Universitat de Lleida

Escola Politècnica Superior

Enginyeria Tècnica en Informàtica de Gestió

Treball Final de Carrera

Disseny i implementació d'una aplicació T.P.V. destinada al petit comerç

Autor: **Marc Freixes Sanjuan**

Codirectors: **Francesc Argilés Pons i Marga Moltó Aribau**

Juny 2013

AGRAÏMENTS

Al meu gran amic, company de feina i codirector d'aquest projecte Francesc Argilés, pel seu inestimable recolzament i implicació en totes les etapes incloses en la consecució del treball final de carrera. Puc afirmar que va ser la persona que em va donar l'empenta que necessitava per decidir-me. Moltes gràcies per tot Frans, t'estaré eternament agraït.

A la directora del projecte Marga Moltó, que tan bon punt va tenir noticia d'aquest projecte no va dubtar ni un segon en acceptar-lo i animar-me en la seva realització. Gràcies Marga, per totes les hores dedicades.

A tots els coneguts, amics i familiars, que no han parat de recordar-me la importància d'aquest projecte i de donar-me ànims en tot moment per no decaure i seguir pel bon camí.

I per últim i en especial als meus pares Pilar i Joan, per la seva incombustible insistència durant tots aquests anys en que no m'oblidés del treball final de carrera. Finalment ho hem aconseguit, moltes gràcies per tot.

Marc Freixes Sanjuan

ÍNDEX DE CONTINGUTS

ÍNDEX DE TAULES

ÍNDEX D'IL·LUSTRACIONS

1 - INTRODUCCIÓ

En el petit comerç i el seu entorn, es produeixen un seguit d'accions totes elles relacionades d'una manera o d'una altra, amb la venda d'articles als clients finals, començant per l'entrada dels articles comprats als proveïdors, passant per l'etiquetatge i acabant per la pròpia venda i el control de vendes posterior a partir de llistats.

Aquest projecte va adreçat a tot el petit negoci que necessita vendre productes d'una manera ràpida, àgil i eficient, així com portar una gestió dels aspectes més rellevants relacionats amb la venda. És en aquest escenari on sorgeix la idea de COMTPV.

COMTPV és el nom que li hem donat a l'aplicació de software de gestió que es desenvolupa en aquest projecte. Aquesta denominació s'extreu de la següent descripció: software T.P.V. [1] per al petit COMerç, per tant a partir d'ara hi farem referència com a COMTPV.

La finalitat de l'aplicació COMTPV serà la de cobrir les necessitats dels petits comerços per tal de poder informatitzar la venda d'articles als clients finals i les gestions relacionades. Per tant s'han de diferenciar dues parts dins de l'aplicació: la pròpia venda a la que tindran accés els venedors del comerç i la resta d'administracions i gestions relacionades a les que només hi podran accedir els administradors de l'empresa.

Totes les opcions, funcions i processos inclosos en l'operativa de COMTPV estaran pensats en facilitar permanentment les tasques a l'usuari final, intentant que siguin el més ràpides, senzilles, àgils i eficients possibles.

En aquesta memòria del TFC es realitzarà un anàlisi exhaustiu de totes les fases necessàries per al correcte desenvolupament i implantació de l'aplicació. A continuació faré una breu menció d'aquestes fases.

En primer lloc, analitzarem el **model de negoci** tot detallant l'activitat de l'empresa a qui va adreçada la nostra aplicació. També hi especificarem els elements que en formaran part.

Tot seguit, en el **procés de desenvolupament de software** definirem els diferents mètodes de desenvolupament que existeixen. Hi explicarem també l'elecció del mètode escollit per al desenvolupament del nostre software COMTPV.

A continuació, mitjançant **l'anàlisi de requeriments i especificació del sistema** descriurem el funcionament del sistema, els requisits funcionals i els requisits no funcionals.

Acte seguit, en l'apartat de la **gestió del risc** analitzarem els diferents tipus de riscos que ens podem trobar en la nostra aplicació. També hi realitzarem les estimacions i gestions conseqüents.

En la **planificació i pressupost** detallarem la distribució anticipada en el temps de les tasques a desenvolupar i les seves despeses derivades.

[1] És l'acrònim de Terminal Punt de Venda. Fa referència al dispositiu i tecnologies que ajuden a la tasca de gestió d'un establiment comercial de venda al públic. Definició extreta de http://www.wikipedia.org.

Tot seguit definirem l'estructura de dades necessària per emmagatzemar tota la informació introduïda en l'aplicació COMTPV a la **base de dades.**

En aquest punt ja ens podem posar amb el **disseny de l'aplicació,** on documentarem les diferents funcionalitats de l'aplicació i la seva estructura.

Una vegada acabat el disseny de l'aplicació, abordarem la fase **d'implementació.** En aquest apartat hi descriurem el llenguatge de programació escollit i uns petits exemples de codi.

A continuació, en la fase de **proves del software** explicarem les proves realitzades a COMTPV per tal de garantir el seu correcte funcionament.

Finalment, anomenarem les **consideracions** a tenir en compte, les possibles **ampliacions** de COMTPV en el futur i les **conclusions** finals de la realització d'aquest projecte.

2 – OBJECTIUS

Es necessari que abans de començar amb el disseny de l'aplicació, fixem quins seran els objectius que s'hauran de complir per a la satisfacció final del client que és l'objectiu més important.

Per tal de poder establir aquests objectius de manera adequada i objectiva, és primordial acordar les reunions necessàries amb el client. D'aquesta manera, podrem observar de primera mà la seva metodologia de treball i les seves necessitats. Així, la nostra aplicació proporcionarà les funcionalitats de manera personalitzada i podrem definir-les adequadament a les necessitats del client i complir els objectiu satisfactòriament.

En el nostre cas els objectius fixats són els següents:

- Gestionar la **compra/venda** d'articles a la botiga de manera ràpida, àgil i eficient.

- Desenvolupar una aplicació que sigui el més **intuïtiva** i fàcil d'utilitzar possible. Aquest punt és força important ja que els venedors no sempre disposaran de la formació informàtica adequada.

- Assegurar la màxima **fiabilitat** de l'aplicació, havent-li aplicat els tests de proves necessaris. A més a més, la dotarem d'una interfície gràfica moderna i atractiva al gust del client.

- La correcta **implantació**, funcionament i formació de l'aplicació a la botiga obtenint la satisfacció del client.

- L'objectiu final serà el de garantir que la nostra aplicació sigui de la millor **qualitat** possible i que superi les **expectatives** de l'usuari final.

3 – MODEL DE NEGOCI

En aquest apartat passarem a descriure l'activitat que desenvolupa l'empresa a la que va dirigida la nostra aplicació de software.

És de vital importància conèixer l'activitat i el funcionament de l'empresa per tal de poder identificar qualsevol factor que pugui influir en algun punt del desenvolupament de l'aplicació.

L'activitat de l'empresa consisteix en la compra/venda de petits articles. En primer lloc, és realitza la compra d'articles a diferents proveïdors. Una vegada es reben els articles comprats, es col·loquen estratègicament a les diferents prestatgeries de la botiga. En cas de rebre una gran quantitat d'unitats d'algun article en concret, aquestes es guarden en un petit magatzem per així més endavant poder restablir els prestatges de la botiga. Finalment els clients entren a la botiga, trien els articles que els interessen i passen per caixa, d'aquesta manera s'acaba efectuant la venda dels articles.

3.1 – ANÀLISI DEL MODEL

En l'apartat anterior hem definit el funcionament de l'empresa d'una forma més genèrica. Tot seguit en l'anàlisi del model, en realitzarem una descripció molt més detallada dels elements que formaran part del procés de desenvolupament de la nostra aplicació.

- **Proveïdors:** s'encarreguen de subministrar tots els articles demanats a la botiga. La majoria d'aquests proveïdors són en realitat grans distribuïdors de productes. La informació més rellevant dels proveïdors és la següent:

  - Nom
  - NIF
  - Telèfons
  - Codi Postal
  - Població
  - Província
  - Adreça
  - E-Mail (adreça de correu electrònic)
  - Lloc Web
  - Persona de contacte
  - Percentatges de descomptes habituals
  - Observacions

- **Articles:** són un element bàsic i fonamental en la nostra aplicació. Els articles sempre pertanyeran a una família i un proveïdor. Les dades que ens interessarà registrar són:

  - Descripció
  - Família
  - Proveïdor
  - Percentatge de marge de benefici en la venda
  - Últim cost (últim preu de compra)
  - Data revisió del últim cost
  - Imatge
  - Observacions
  - Períodes d'ofertes
    - Data inici
    - Data fi
    - Preu
    - Descompte

- **Famílies:** es tracta d'un element útil per la classificació i definició de característiques genèriques per un conjunt d'articles. En guardarem les següents dades:

  - Descripció
  - Percentatge de marge de benefici en la venda

- **Clients:** qualsevol persona que entri a la botiga i se li realitzi la venda d'algun article. En registrarem la següent informació:

  - Nom
  - NIF/CIF
  - Telèfons
  - Codi Postal
  - Població
  - Província
  - Adreça
  - Observacions

- **Venedors:** seran els treballadors de la botiga autoritzats per poder vendre articles a caixa. Ens interessarà enregistrar els atributs:

  - Nom
  - Observacions

- **Caixes TPV:** les diferents localitzacions dins la botiga on els clients podran efectuar el pagament de les seves compres. Ens serviran per pre-configurar algunes característiques. Necessitarem guardar la següent informació:

  - Descripció
  - Venedor associat per defecte a la caixa
  - Client de comptat associat per defecte a la caixa
  - Observacions

- **Usuaris:** persones autoritzades a entrar en la nostra aplicació. Depenent del tipus d'usuari, accedirem a l'aplicació amb accés total o únicament a les vendes. Atributs necessaris:

  - Nom
  - Clau (password)
  - Tipus (administrador / venedor)

- **Document de compra:** document per tal d'informatitzar la compra d'articles als nostres proveïdors. Hi podrem veure la següent informació:

  - Capçalera
    - Número de document
    - Data de la compra
    - Proveïdor
      - Codi
      - Nom
      - NIF/CIF
      - Telèfon
      - Codi Postal
      - Població
      - Adreça
    - Observacions

  - Detall
    - Article
    - Descripció de l'article
    - Unitats
    - Preu unitari
    - Descompte 1
    - Descompte 2
    - Import

  - Peu
    - Total (suma dels imports de totes les línies)

- **Document de venda:** és la peça clau de la nostra aplicació, ens ha de garantir una venda ràpida, àgil i eficient. Es tracta del formulari que estarà sempre actiu a les caixes de la botiga, i mitjançant el qual es realitzarà la venda d'articles als clients finals. Hi podrem veure/introduir les dades següents:

  - Capçalera
    - Número de document
    - Data-Hora de la venda
    - Caixa
    - Venedor
    - Client
      - Codi
      - Nom
      - NIF/CIF
      - Telèfon
      - Codi Postal
      - Població
      - Adreça
    - Observacions

  - Detall
    - Article
    - Descripció de l'article
    - Unitats
    - Preu unitari
    - Descompte
    - Import

  - Peu
    - Total (suma dels imports de totes les línies)

- **Còpies de seguretat:** guardarem un històric de totes les còpies de seguretat realitzades pels usuaris amb la següent informació:

  - Data-hora de la còpia de seguretat
  - Fitxer on s'ha guardat la còpia

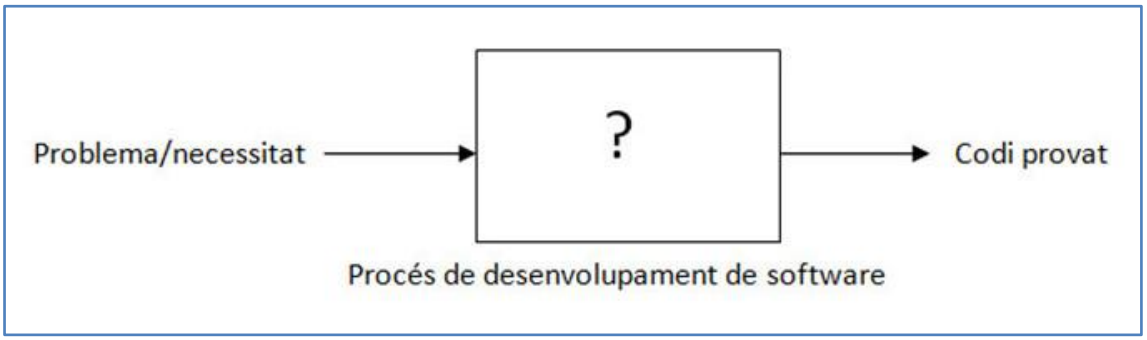
4 – PROCÉS DE DESENVOLUPAMENT DE SOFTWARE

Abans de posar-nos de ple amb la definició del procés de desenvolupament de software, és necessari presentar una de les moltes definicions de software que existeixen.

El software d'ordinador és el producte que dissenyen i construeixen els enginyers de software. Això avarca programes que s'executen dins d'un ordinador de qualsevol mida i arquitectura, documents que contenen formularis virtuals i impresos, dades que combinen números i text i també inclouen representacions d'informació d'àudio, vídeo i imatges [PRE02].

Un procés de desenvolupament de software és una estratègia per desenvolupar, provar i mantenir una aplicació de software [PRE92].

Normalment es comença amb un problema o necessitat al que apliquem un procés de desenvolupament de software i finalment acabem obtenint un codi provat.



Il·lustració 1. Procés de desenvolupament de software

El procés de desenvolupament de software està format per les següents fases :

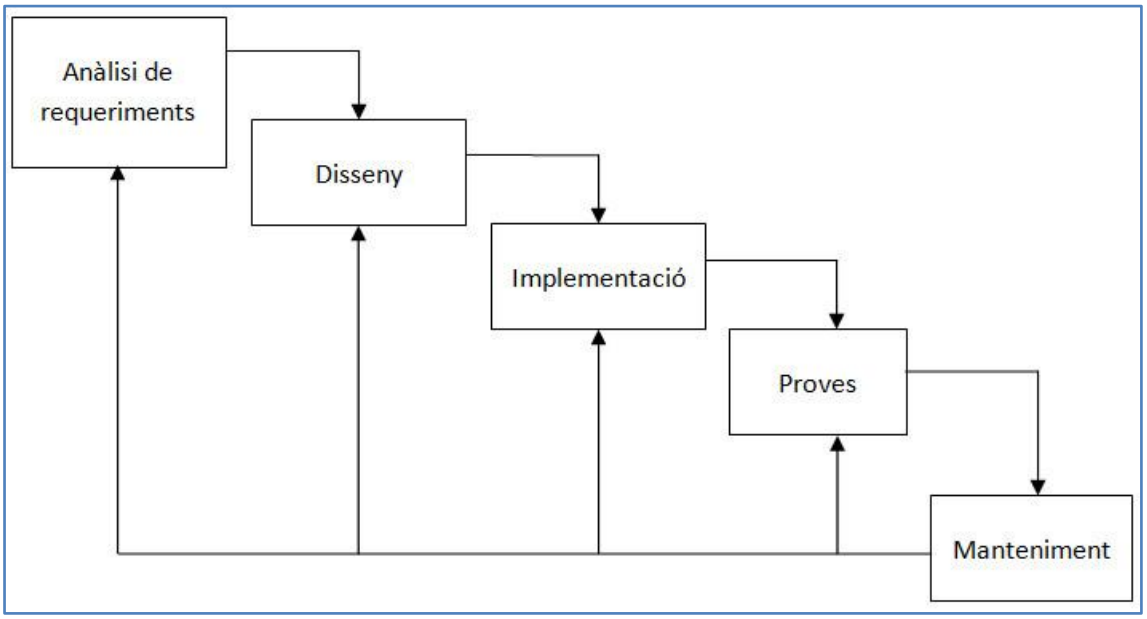
- Anàlisi de requeriments
  - Comprensió del problema
  - Especificació dels requeriments
- Disseny
  - Disseny preliminar (detecció de mòduls)
  - Disseny detallat (lògica interna de cada mòdul)
- Implementació
- Proves
  - Proves d'unitat
  - Proves d'integració
  - Proves de sistema
  - Proves de validació
- Manteniment

4.1 – MÈTODES DE DESENVOLUPAMENT

Aquests mètodes, a partir d'una sèrie de models, notacions, criteris, recomanacions de disseny i processos base, ens guiaran en la creació de software a nivell tècnic.

4.1.1 – MODEL SEQÜENCIAL

Model també anomenat lineal, en cascada, cicle de vida clàssic o salt d'aigua. Aquest model ens suggereix un enfocament sistemàtic i seqüencial per al desenvolupament del software.



Il·lustració 2. Model seqüencial

Perquè falla alguns cops el model seqüencial:

- Els projectes reals poques vegades segueixen el model seqüencial que ens proposa aquest model. Encara que el model seqüencial permet certa interacció, ho fa indirectament i com a resultat els canvis poden causar confusió.

- Sovint, és difícil que el client exposi explícitament tots els requeriments. Aquest model ho requereix i té dificultats a l'hora d'albergar certa incertesa al inici dels projectes.

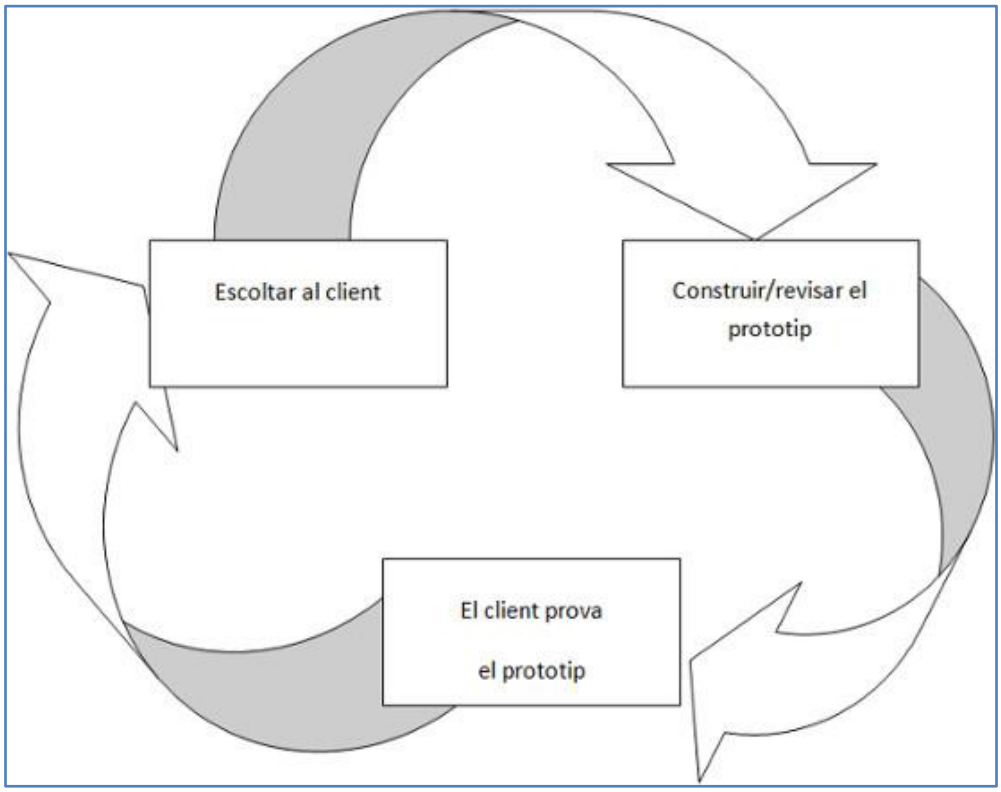
- El client ha de tenir molta paciència, ja que una versió de treball del programa no estarà disponible fins a fases molt avançades del projecte. Un error greu pot resultar desastrós si no és detecta fins que es revisa el programa.

Cada un d'aquests errors és real, però tot i això el model seqüencial té un lloc important en la enginyeria del software, donat que proporciona una plantilla en la que hi trobem mètodes per l'anàlisi, el disseny, la implementació, les proves i el manteniment. Aquest model segueix sent un dels més utilitzats a pesar dels inconvenients.

4.1.2 – MODEL DE CONSTRUCCIÓ DE PROTOTIPS

El client sovint defineix un conjunt d'objectius generals per al software, però no identifica els requeriments inicials, de procés o sortida. En altres casos el responsable del desenvolupament del software, pot no estar segur de la eficàcia d'un algoritme, de la capacitat d'adaptació d'un sistema operatiu o de la forma en que s'hauria de tractar la interacció persona-ordinador. En aquestes i moltes altres situacions, el model de construcció de prototips pot oferir-nos el millor enfocament.

Aquest model comença amb la recol·lecció de requeriments. El desenvolupador i el client es reuneixen i defineixen els objectius globals per al software, identifiquen els requisits coneguts i les àrees del esquema on és obligatòria més definició. En aquest punt apareix un disseny ràpid que es centra en la representació d'aquests aspectes que seran visibles per l'usuari/client. El disseny ràpid ens porta a la construcció d'un primer prototip que avaluarà el client i s'utilitzarà per refinar els requeriments del software a desenvolupar.



Il·lustració 3. Model de construcció de prototips

És cert que tant als usuaris/clients com als desenvolupadors del software els agrada el model de construcció de prototips. Als clients els atreu el sistema real i als desenvolupadors els perd l'idea de començar a construir alguna cosa immediatament. No obstant, la construcció de prototips també ens pot ocasionar els següents inconvenients:

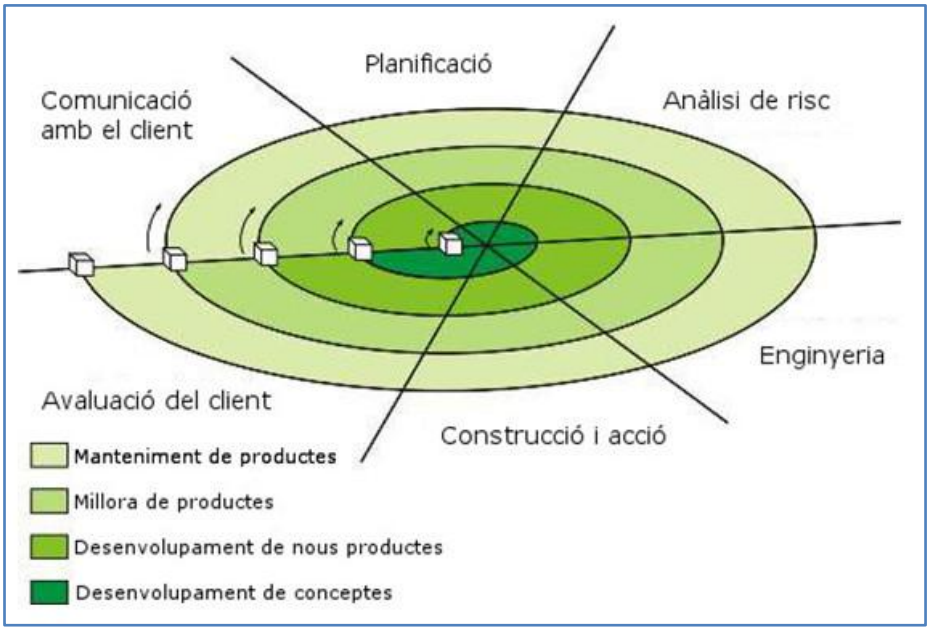
- El client veu una versió de treball del software sense tenir coneixement que amb les presses per fer que funcioni, no s'ha tingut en compte la qualitat del software global o la facilitat de manteniment a llarg termini. Quan s'informa al client que s'ha de tornar a construir el software per poder garantir un alt nivell de qualitat, no ho entén i demana que es finalitzi amb uns petits retocs.

- El desenvolupador, freqüentment, fa compromisos d'implementació per fer que el prototip funcioni ràpidament. És pot utilitzar un sistema operatiu o un llenguatge de programació inadequat simplement perquè són coneguts.

Encara que poden sorgir problemes, la construcció de prototips pot ser un model efectiu. La clau està en la definició de les regles del joc al inici. El client i el desenvolupador s'han de posar d'acord en que el prototip es construeix per utilitzar-lo com a mecanisme per a la definició de requeriments.

4.1.3 – MODEL EN ESPIRAL

El model en espiral, proposat originalment per Boehm [BOE88], és un model de procés de software evolutiu que conjuga la naturalesa iterativa de construcció de prototips amb els aspectes controlats i sistemàtics del model seqüencial. Proporciona el potencial per un desenvolupament ràpid de les versions incrementals del software. Durant les primeres iteracions, la versió incremental podria ser un model de paper o un prototip. A les últimes iteracions es produeixen versions cada vegada més complertes del sistema dissenyat.



Il·lustració 4. Model en espiral

El model en espiral es divideix en sis tasques:

- Comunicació amb el client
- Planificació
- Anàlisi de risc
- Enginyeria
- Construcció i acció
- Avaluació del client

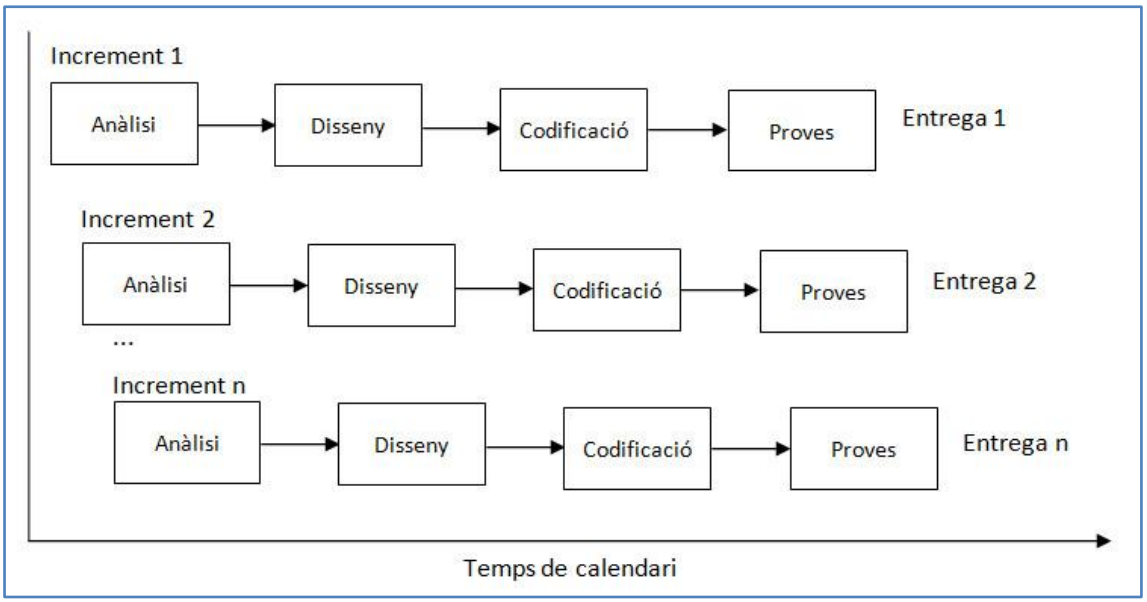
A diferència del model de procés clàssic que acaba quan s'entrega el software, el model en espiral pot adaptar-se i aplicar-se al llarg de la vida del software.

Al igual que altres models, el model en espiral tampoc es perfecte. Pot resultar difícil convèncer a grans clients, sobretot sota contracte, de que l'enfocament evolutiu es controlable. Requereix una considerable habilitat per la avaluació del risc, sent aquesta gran part de l'èxit. Si un risc no es descobreix i gestiona a temps, sorgiran problemes greus.

4.1.4 – MODEL INCREMENTAL

El model incremental combina elements del model seqüencial amb la filosofia interactiva de la construcció de prototips. Consisteix en aplicar seqüències lineals de forma escalonada mentre es va avançant en el temps de calendari. Cada seqüència lineal produeix un "increment" en el software, en el que s'afegeixen noves funcionalitats o se'n milloren d'existents. S'ha de tenir en compte que durant qualsevol increment és possible incorporar la construcció de prototips.

Quan s'utilitza aquest model, el primer increment acostuma a ser un producte essencial. És a dir, es codifiquen requisits bàsics però moltes funcions suplementàries queden pendents per als següents increments. A diferència del model de construcció de prototips, el model incremental es centra en l'entrega d'un producte operacional a cada increment.



Il·lustració 5. Model incremental

4.2 – MODEL DE PROCÉS UTILITZAT

Una vegada analitzats tots els pros i contres dels models de procés descrits anteriorment, he decidit decantar-me pel **model incremental**.

Si ens posem en el rol de desenvolupador com en el de client, el fet que a cada increment es pugui realitzar una versió operativa per al client a partir d'un anàlisi de requeriments i fins i tot amb la construcció d'un prototip em sembla ideal. D'aquesta manera el client podrà disposar d'una primera versió (increment) operativa amb certa funcionalitat bàsica, i a partir d'aquesta primera versió, anar rectificant possibles operatives i afegint noves funcionalitats fins obtenir una versió que s'adeqüi íntegrament a les seves necessitats.

Tenir en compte que amb l'aplicació d'aquest model, el software no quedarà mai tancat, sempre estarà obert a nous increments. Aquest fet permetrà al nostre client la possibilitat de futures ampliacions en la nostra aplicació.

5 – ANÀLISI DE REQUERIMENTS I ESPECIFIACIÓ DEL SISTEMA

L'anàlisi de requeriments és una tasca d'enginyeria del software que es troba entre la definició del software a nivell de sistema i el disseny del software. L'anàlisi de requeriments permet a l'enginyer informàtic especificar les característiques operacionals del software (funció, dades i rendiment), la interfície amb altres elements del sistema i establir les restriccions que ha de complir el software.

S'ha d'entendre els requeriments d'una forma dinàmica, ja que és molt probable que vagin canviant durant el desenvolupament del software. La mesura en la que ens anem adaptant als canvis sorgits en els requeriments, anirà lligada a l'èxit final del nostre desenvolupament.

L'anàlisi de requeriments del software pot dividir-se en cinc àrees d'esforç:

1. **Reconeixement del problema**: identificar els elements bàsics del problema tal i com els percep el client.
2. **Avaluació i síntesi**: definir tots els objectes de dades observats externament, avaluar el flux i contingut de la informació, definir i elaborar totes les funcions del software, entendre el comportament del software en el context del sistema, establir la interfície del sistema i descobrir restriccions addicionals al disseny.
3. **Modelatge**: creació de models del sistema per tal d'entendre millor el flux de dades i de control, el tractament funcional, el comportament operatiu i el contingut de la informació.
4. **Especificació**: a partir dels models creats en l'àrea anterior es procedirà a l'especificació del software. Consisteix en definir els objectius del software descrivint-lo en el context del sistema basat en l'ordinador.
5. **Revisió**: revisió tant per part del desenvolupador com del client de les especificacions definides, per tal de descobrir algun problema que hagi passat desapercebut en primera instancia.

Sovint, els requeriments de sistemes de software es classifiquen en funcionals i no funcionals, tot seguit en dediquem dos apartats per explicar-los.

5.1 – REQUERIMENTS FUNCIONALS

Els requeriments funcionals són tots aquells que ens descriuen el comportament desitjat del software. Una altra manera de veure-ho, seria com totes aquelles funcions requerides per tal de solucionar el problema.

Ens proporcionen una descripció del procés de cada funció, s'estableixen i justifiquen les restriccions del disseny, es defineixen les característiques del rendiment i s'inclouen un o més diagrames per representar gràficament l'estructura global del software i les interaccions entre les seves funcions i altres elements del sistema.

Cada requisit funcional ens mostra la relació entre les entrades i les sortides del sistema, tot tenint en compte les possibles excepcions.

A partir de les primeres reunions realitzades amb el client, hem observat i diferenciat els següents requeriments funcionals que haurà de complir la nostra aplicació:

- Gestionar la **venda** d'articles a la botiga de manera ràpida, àgil i eficient. Aquest apartat inclou l'alta, la baixa, la modificació, la consulta i la impressió del document de venda. També incorpora la navegació a tots els elements relacionats a la venda(client, venedor, articles,...).

- Tramitar la **compra** d'articles a proveïdors per la nostra botiga. Inclou l'alta, la baixa, la modificació i la consulta de documents de compra. Podrem també navegar a tots els elements relacionats a la compra(proveïdor, articles,...).

- Portar una gestió dels **clients** introduïts en les vendes. Cal prestar atenció que majoritàriament s'utilitzarà el mateix client (client de comptat) a l'hora de vendre, i que únicament en els clients habituals serà en els que obrirem fitxes personalitzades. Hi podrem accedir per donar d'alta, de baixa, modificar o consultar clients.

- Administrar els **venedors** relacionats en les vendes, amb la possibilitat de realitzar les operacions d'alta, de baixa, de modificació o de consulta.

- Configurar les **caixes T.P.V** utilitzades en les vendes amb les següents operacions disponibles: alta, baixa, modificació i consulta. Tenir en compte que en aquest apartat hi podrem indicar el venedor i el client de comptat que ens apareixeran per defecte en el moment de realitzar la venda.

- Gestionar els **proveïdors** escollits en les compres podent efectuar les operacions d'alta, de baixa, de modificació o de consulta

- Manteniment dels **articles** introduïts en les compres i les vendes amb la possibilitat de portar a cap les operacions d'alta, de baixa, de modificació, de consulta o d'impressió de l'etiqueta amb el codi de barres EAN[2]. També hi podrem establir períodes d'ofertes.

- Introducció de les **famílies** relacionades amb els articles. Ens seran útils en els posteriors llistats de vendes. Hi podrem complir les operacions d'alta, de baixa, de modificació o de consulta.

- Realitzar la **impressió de les etiquetes** dels articles a vendre amb els seus codis de barres per tal de facilitar l'etiquetatge i posterior venda.

- Proporcionar **llistats de les vendes** efectuades podent parametritzar-los amb diversos filtres i opcions.

[2] *European Article Number (EAN)* és un sistema de codi de barres adoptat per més de 100 països i prop d'un milió d'empreses en l'any 2003. Informació extreta de http://www.wikipedia.org.

- Administrar els **usuaris** utilitzats en la nostra aplicació permetent efectuar les operacions d'alta, de baixa, de modificació o de consulta. Comentar que disposarem de dos tipus d'usuaris: administrador o venedor.

- Gestionar les **còpies de seguretat** de totes les dades introduïdes en la nostra aplicació. Ens permetrà realitzar còpies de seguretat i poder restaurar-les.

5.1.1 – ESPECIFICACIÓ DE CASOS D'ÚS

Una de les millors formes de mostrar els requeriments funcionals de la nostra aplicació és mitjançant els casos d'ús. Podríem definir cas d'ús com un o més escenaris que ens indiquen la interacció entre el sistema i els usuaris per tal d'aconseguir algun dels objectius dels usuaris.

A partir dels casos d'ús, mostrarem els possibles escenaris en que es podrà trobar l'usuari en l'ús de la nostra aplicació, així com la forma d'interactuar amb ella.

A continuació anem a especificar alguns dels casos d'ús més significatius del nostre software, la resta de casos d'ús els podem trobar en l'annex de l'apartat 16.1.

Cas d'ús 1. Alta d'article / família / client / proveïdor / venedor / caixa TPV/ usuari

He ajuntat tots els processos d'alta en un sol cas d'ús, ja que la operativa és exactament la mateixa, només canvien els atributs.

En aquest cas d'ús es dóna d'alta un article / família /client / proveïdor / venedor / caixa TPV / usuari amb els atributs corresponents.

Atributs article: codi, descripció, família, proveïdor, imatge, % marge venda, data/hora última revisió cost, últim cost, observacions i ofertes.

Atributs família: codi, descripció i % marge venda.

Atributs client: codi, nom, NIF, telèfon, població, adreça, codi postal, província i observacions.

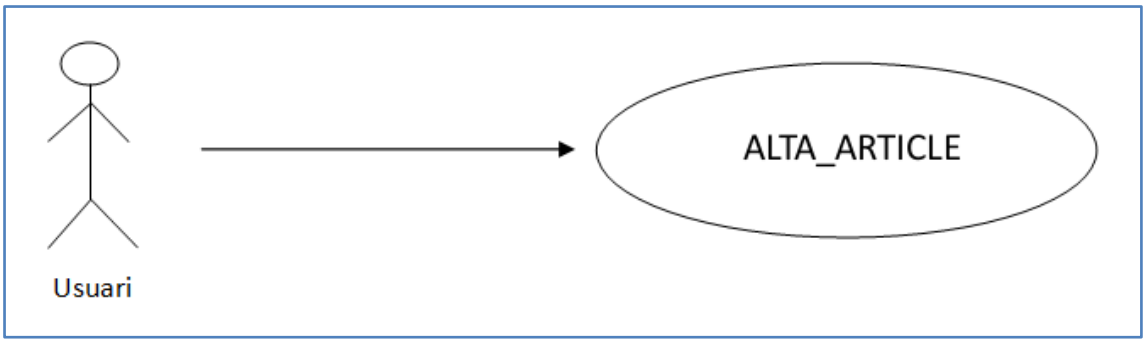
Atributs proveïdor: codi, nom, NIF, telèfon, població, adreça, codi postal, província, descompte1, descompte2, email, web, contacte i observacions.

Atributs venedor: codi, nom i observacions.

Atributs caixa TPV: codi, descripció, venedor, client de comptat i observacions.

Atributs usuari: nom, clau i tipus (administrador, venedor).

Cas d'ús "Alta_article"



Il·lustració 6. Cas d'ús "Alta_article"

Flux d'esdeveniments "Alta_article"

Usuari	Sistema
1.L'usuari vol donar d'alta un article, pitja el botó "Articles".	
	2.El sistema mostra el formulari per al manteniment d'articles inhabilitat amb diferents botons per les accions a realitzar.
3.L'usuari pitja el botó "Nou"	
	4.El sistema ens dóna un nou codi d'article automàticament i habilita totes les caselles del formulari perquè l'usuari pugui introduir els atributs del nou article.
5.L'usuari introdueix els atributs desitjats per al nou article.	
6.L'usuari pitja el botó "Acceptar" per confirmar els atributs introduïts.	
	7.El sistema comprova els atributs introduïts i si tot es correcte emmagatzema els atributs del nou article a la base de dades.
	8.En cas que hi hagi algun atribut incorrecte el sistema mostra un missatge d'avís a l'usuari.

Taula 1. Flux d'esdeveniments "Alta_article"

PRE: {L'usuari vol crear un nou article.}

POST: {S'ha creat un nou article amb els atributs introduïts.}

EXCEPCIÓ: {El codi d'article introduït ja existeix.}

En el cas d'ús "Alta_article", cal comentar que en la pròpia pantalla d'articles disposarem d'accés al manteniment de famílies per poder-ne donar d'alta en cas necessari. A més també hi podrem establir els períodes de les ofertes relacionades amb l'article.

Cas d'ús 2. Venda d'articles

En aquest cas d'ús es realitza la venda d'articles, eina principal de l'aplicació que ens servirà per vendre els articles als clients de forma ràpida, àgil i eficient. Per tal de poder realitzar una venda ràpida de cara al públic el document de venda tindrà parts configuratives com el codi de venedor, el codi de client comptat per a la majoria de vendes a clients anònims i el codi de caixa activa. D'aquesta manera, al fer una nova venda ja anirem directament a la introducció del detall.

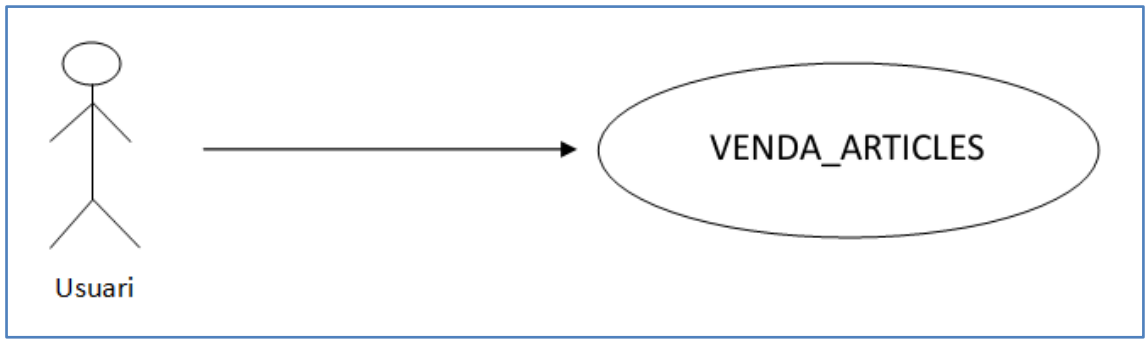
Cabria una segona possibilitat on realitzaríem una venda a un client conegut i codificat a la base de dades. En aquest cas l'usuari hauria de modificar la capçalera del document escollint el client adequat.

L'usuari tindria la possibilitat d'introduir els articles mitjançant un lector de codis de barres de formà ràpida. En aquest cas introduiria el codi d'article amb una unitat i calculant l'import automàticament, també es posicionaria a la següent línia per més comoditat.

Una altra característica interessant, és que els usuaris de tipus "Venedor" al entrar a l'aplicació ja aniran directament a aquest formulari de venda d'articles.

Els atributs d'un document de venda són el codi de caixa, el codi de client, el codi de venedor, el número de document, la data i els diferents codis d'articles amb les seves unitats, preus i descomptes respectius.

Cas d'ús "Venda_articles"



Il·lustració 7. Cas d'ús "Venda_articles"

Flux d'esdeveniments "Venda_articles"

Usuari	Sistema
1.L'usuari es troba de cara al públic i vol realitzar la venda ràpida d'articles, pitja el botó "Document de venda(T.P.V.)". En cas d'usuaris de tipus venedor, ja tindran el document de venda obert en pantalla.	

	2.El sistema mostra el formulari per a la venda d'articles amb la capçalera ja omplerta amb la caixa TPV activa, el client de comptat i el venedor de la caixa activa, el número de document de venda i la data actual. Ja es posiciona al detall per la introducció d'articles.
3.L'usuari va introduint les diferents línies del detall amb el codi d'article, les unitats, el preu i el descompte. Té la possibilitat d'utilitzar un lector de codi de barres per més rapidesa.	
	4.El sistema posarà per defecte una unitat del article introduït, el preu de venda calculat a partir de l'últim cost + el % de marge de la fitxa de l'article i calcularà l'import a cada línia i els totals del document. En cas que l'article introduït tingui alguna oferta definida a la seva fitxa, s'obtindrà el preu i descomptes definits en l'oferta.
5.En el moment que l'usuari hagi introduït totes les línies i el document ja estigui complert pitjarà el botó "Acceptar". En cas que volgués desfer tot el document pitjaria el botó "Cancel·lar ".	
	6.El sistema comprova els atributs introduïts i si tot es correcte emmagatzema els atributs del nou document de venda a la base de dades.
	7.En cas que hi hagi algun atribut incorrecte el sistema mostra un missatge d'avís a l'usuari.

Taula 2. Flux d'esdeveniments "Venda_articles"

PRE: {L'usuari vol vendre articles a un client anònim. Els articles, el client comptat, el venedor per defecte i la caixa activa han d'existir a la base de dades.}

POST: {S'ha generat un nou document de venda.}

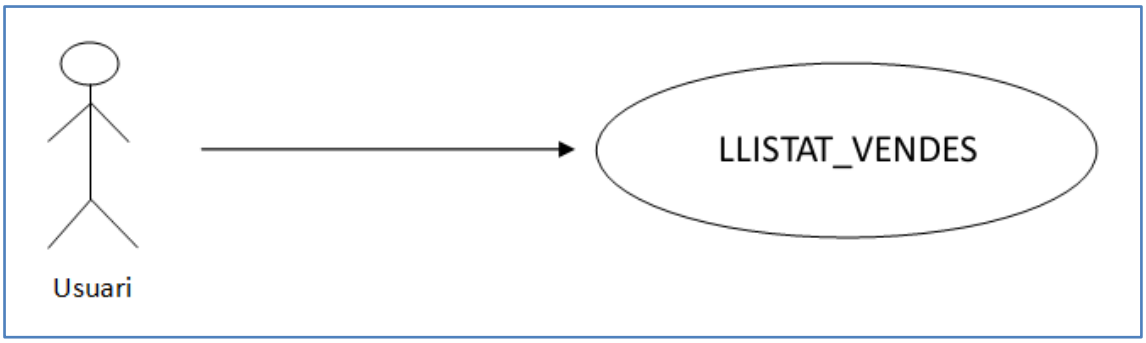
EXCEPCIÓ: {}

Cas d'ús 3. Llistat de vendes

En aquest cas d'ús es realitza un llistat de les vendes efectuades.

Serà de molta utilitat per l'usuari poder llistar les vendes realitzades segons diferents criteris de filtre i agrupació.

Cas d'ús "Llistat_vendes"



Il·lustració 8. Cas d'ús "Llistat_vendes"

Flux d'esdeveniments "Llistat_vendes"

Usuari	Sistema
1.L'usuari vol realitzar un llistat de vendes, pitja el botó "Llistat de vendes"	
	2.El sistema mostra el formulari per al llistat de vendes amb diferents criteris de filtres i agrupació de les dades.
3.L'usuari introdueix els filtres i criteris desitjats i pitja el botó "Llistar".	
	4.El sistema consulta la informació a la base de dades i mostra el llistat per pantalla.
5.L'usuari té la possibilitat d'imprimir el llistat que està visualitzant. Pitja el botó "Imprimir".	
	6.El sistema envia el llistat per impressora.

Taula 3. Flux d'esdeveniments "Llistat_vendes"

PRE: {L'usuari vol realitzar un llistat de vendes.}

POST: {S'ha realitzat un llistat de vendes.}

EXCEPCIÓ: {No hi ha vendes a mostrar segons els criteris escollits per l'usuari. Errors relacionats amb la impressora.}

5.2 – REQUERIMENTS NO FUNCIONALS

Els requeriments no funcionals són aquells requeriments que no es refereixen directament a les funcions específiques que proporciona el sistema, sinó a les propietats emergents d'aquest com la fiabilitat, el temps de resposta i la capacitat d'emmagatzemament. De forma alternativa, defineixen les restriccions del sistema com la capacitat dels dispositius d'entrada/sortida i les representacions de dades que s'utilitzen en les interfícies del sistema.

Aquest tipus de requeriments, no només es refereixen al sistema de software a desenvolupar, alguns poden restringir el procés que s'ha d'utilitzar per desenvolupar el sistema.

De vegades, els requeriments no funcionals sorgeixen de les necessitats de l'usuari, degut a les restriccions en el pressupost, a les polítiques de la organització, a la necessitat de compatibilitat amb altres sistemes de software o hardware, o a factors externs com regulacions de seguretat o privacitat.

Els tipus de requeriments no funcionals són:

1. **Requeriments del producte:** especifiquen el comportament del producte.

    a. Requeriments de rendiment en la rapidesa d'execució del sistema i quanta memòria requereix.
    b. Requeriments de fiabilitat que fixen la taxa d'errades per a que el sistema sigui acceptable.
    c. Requeriments de portabilitat.
    d. Requeriments d'usabilitat per facilitar l'ús del sistema a l'usuari.

2. **Requeriments organitzacionals:** es deriven de polítiques i procediments existents en les organitzacions del client i del desenvolupador.

    a. Estàndards en els processos a utilitzar.
    b. Requeriments d'implementació com els llenguatges de programació o el mètode de disseny a utilitzar.
    c. Requeriments d'entrega.

3. **Requeriments externs:** aquest gran apartat inclou tots els requeriments que deriven de factors externs al sistema i del seu procés de desenvolupament.

    a. Requeriments per tal d'adquirir la compatibilitat amb sistemes d'altres organitzacions.
    b. Requeriments legislatius per assegurar el compliment de les lleis que puguin afectar al producte.

5.3 – ESPECIFICACIÓ DEL SISTEMA

En aquest apartat definirem els requeriments mínims i els recomanats perquè l'aplicació funcioni correctament. També descriurem tot el software utilitzat en el desenvolupament de l'aplicació.

5.3.1 – REQUERIMENTS MÍNIMS

Els requeriments mínims són tot aquell conjunt de hardware i software necessari perquè l'aplicació es pugui executar sense cap inconvenient.

En el nostre software els requeriments mínims són els següents:

Requeriments mínims	
Sistema operatiu	Microsoft Windows XP
Servidor de dades	Microsoft SQL Server 2005 Express
Components de software	Microsoft .NET Framework 4.0
Monitor	17" amb resolució de 1024 x 768
Processador	1 GHz
Memòria RAM	512 MB
Disc dur	2 GB
Perifèrics	Ratolí, teclat i impressora estàndards

Taula 4. Requeriments mínims

5.3.2 – REQUERIMENTS RECOMANATS

Els requeriments recomanats són tot aquell conjunt de hardware i software necessari perquè l'aplicació es pugui executar amb tota la rapidesa i operativitat possible.

En el nostre software els requeriments recomanats són els següents:

Requeriments recomanats	
Sistema operatiu	Microsoft Windows 7
Servidor de dades	Microsoft SQL Server 2008
Components de software	Microsoft .NET Framework 4.0
Monitor	23" amb resolució de 1680 x 1050
Processador	2,33 GHz
Memòria RAM	2 GB
Disc dur	10 GB
Perifèrics	Ratolí, teclat i impressora estàndards, impressora d'etiquetes i lector de codis de barres compatible EAN

Taula 5. Requeriments recomanats

5.3.3 – SOFTWARE UTILITZAT EN EL DESENVOLUPAMENT DE L'APLICACIÓ

Tot seguit es detalla tot el software utilitzat durant el desenvolupament de l'aplicació:

- **Microsoft Windows 7 Ultimate:** Sistema operatiu escollit per la realització de totes les parts del projecte. Alguns del motius d'aquesta elecció són: la gran compatibilitat amb programes i connectors externs, la rapidesa en el processament de dades i el coneixement previ del sistema.

- **Microsoft SQL Server 2008 Express:** és el sistema d'administració de dades triat per emmagatzemar i gestionar les dades de l'aplicació. És tracta d'una edició gratuïta de SQL Server eficaç i de confiança que ens ofereix un variat conjunt de característiques. Cal tenir en compte també que ja disposava de cert coneixement de l'eina.

- **Microsoft Visual C# 2010:** llenguatge de programació utilitzat per la codificació de l'aplicació. Aquest llenguatge es troba inclòs dins del paquet Visual Studio 2010 de Microsoft, i ens ofereix la possibilitat de crear aplicacions d'escriptori.

  Alguns dels fets que m'han fet decantar per aquest llenguatge són els següents:

  - És un llenguatge de programació orientat a objectes simple, robust, eficaç, segur i elegant.
  - Actualment Visual Studio té un dels millors entorns de desenvolupament del mercat.
  - Integració dins la plataforma .NET, això ens permetria realitzar o aprofitar components fets en altres llenguatges d'aquesta plataforma com podria ser Visual Basic.
  - Sembla que des de Microsoft s'aposta molt per aquest llenguatge ja que ha estat creat exclusivament per treballar en la plataforma .NET.
  - Llenguatge totalment nou per mi que m'ajudarà a créixer professionalment.

- **SAP Crystal Reports for Visual Studio 2010:** eina que s'integra a la perfecció dins de l'entorn del Visual Studio. Ens ha servit per dissenyar i generar informes d'una forma molt visual i intuïtiva a partir de les dades emmagatzemades per l'aplicació.

- **Microsoft Office Word 2007:** paquet inclòs dins del Microsoft Office 2007 amb el que he elaborat tota la documentació del projecte. M'he decantat per aquesta eina pel fet de tenir-ne un bon coneixement i de la simplicitat del seu ús.

- **Microsoft Office Power Point 2007:** es tracta d'un altre paquet incorporat al Microsoft Office 2007. En aquest cas ha estat escollit per dissenyar les dispositives que es presentaran en l'exposició del projecte al tribunal.

- **DIA:** per la implementació del model entitat-relació he utilitzat un programari de lliure distribució anomenat DIA que permet realitzar diferents tipus de diagrames tècnics de

forma molt ràpida i instintiva. A més permet la exportació a diferents formats com per exemple .PDF, .JPG o .PNG.

- **GIMP:** software de lliure distribució mitjançant el que fet petits retocs d'imatge al fons i a les icones utilitzades en l'aplicació.

6 – GESTIÓ DEL RISC

Un apartat important en la en la gestió de projectes, és anticipar els riscs que podrien afectar a la programació del projecte o a la qualitat del software a desenvolupar i emprendre accions per evitar aquests riscs. Els resultats d'un anàlisi de riscs s'han de documentar durant el pla del projecte, junt amb l'anàlisi de conseqüències quan el risc succeeix. Identificar els riscs i crear plans per minimitzar els seus efectes sobre el projecte s'anomena gestió de riscs [OUL99].

D'una forma simple, es pot concebre un risc com una probabilitat de que una circumstància adversa succeeixi. Els riscs són una amenaça pel projecte, pel software que s'està desenvolupant i per la organització.

Es podria categoritzar els riscs tal com es mostra a continuació:

- **Riscs de projecte:** aquests afecten a la planificació del calendari o als recursos del projecte. Uns exemples podrien ser la pèrdua d'un dissenyador experimentat o bé un canvi en els requisits inicials del projecte.

- **Riscs de producte:** la qualitat i el rendiment del software que s'està desenvolupant es veu afectat per aquest tipus de risc. Com exemple, podríem anomenar el fet de comprar un component extern del qual hem obtingut un rendiment inferior al esperat. També hauríem de tenir en compte l'ambigüitat en l'especificació del sistema.

- **Riscs de negoci:** afecten a la organització que desenvolupa o subministra el software i posen en perill la viabilitat del projecte. Es desglossen en tres subcategories:

    - **Riscs de mercat:** el fet de crear una aplicació, per molt bona que sigui, sinó té un mercat determinat, no tindrà sortida en el món empresarial. Que un competidor introdueixi al mercat un nou producte, també és un risc de mercat.
    - **Riscs de gestió:** degut a un canvi d'enfocament o de personal, el client pot perdre la confiança en el projecte.
    - **Riscs de pressupost:** sobrepassar l'import inicial del pressupost seria també un risc de negoci.

La gestió de riscs és important particularment per als projectes de software, degut a les incerteses amb les que s'enfronten molts projectes. Aquestes incerteses acostumen a ser conseqüència d'uns requeriments ambigus. Les dificultats en l'estimació dels temps, els recursos per al desenvolupament del software, la dependència de les habilitats individuals i els canvis en els requeriments degut a les necessitats del client, podrien ser les incerteses més comuns.

Es precís anticipar-se als riscs, comprendre molt bé el impacte d'aquests en el projecte, producte i negoci, i considerar els passos per evitar-los. En el cas que finalment acabin succeint, s'hauran de crear plans de contingència per tal que sigui possible aplicar accions de recuperació.

El procés de gestió de riscs està dividit en varies etapes:

1. **Identificació de riscs:** identificar els possibles riscs per al nostre projecte, producte i negoci.
2. **Anàlisi de riscs:** valorar les probabilitats i conseqüències d'aquests riscs.
3. **Planificació de riscs:** crear plans per atacar els riscs, ja sigui per evitar-los o minimitzar els seus efectes al projecte.
4. **Supervisió de riscs:** valorar els riscs de forma constant, i revisar els plans per la mitigació de riscs tan aviat com la informació del risc estigui disponible.

6.1 – IDENTIFICACIÓ DE RISCS

Aquesta és la primera etapa en la gestió de riscs, que alberga el descobriment dels possibles riscs del projecte. En principi, en aquesta etapa no cal valorar els riscs, tot i que en la pràctica és difícil considerar un risc amb conseqüències menors.

La identificació de riscs, es pot realitzar a través d'un procés de grup utilitzant un enfocament de tipus *brainstorming*[3], o simplement poden basar-se en la experiència. Per agilitzar el procés, s'utilitza una llista dels possibles tipus de riscs. Almenys hi ha sis tipus de riscs que poden aparèixer:

1. **Riscs de tecnologia:** es deriven de les tecnologies de software o hardware utilitzades en el sistema que s'està desenvolupant.
2. **Riscs de personal:** riscs associats a les persones de l'equip de desenvolupament.
3. **Riscs de la organització:** s'originen en l'entorn de la organització on el software s'està desenvolupant.
4. **Riscs d'eines:** es deriven de l'ús d'altres softwares en el desenvolupament del sistema.
5. **Riscs de requeriments:** apareixen a partir dels canvis en els requeriments propiciats pels clients.
6. **Riscs d'estimació:** associats a les característiques del sistema i dels recursos requerits per construir-lo.

En el nostre projecte hem considerat els següents riscs:

Tipus	Risc	Descripció
Personal	Desconeixement de les eines de treball	El desenvolupador del projecte no té experiència en el llenguatge de programació utilitzat per realitzar el projecte
Requeriments	Canvis en els requeriments	El client proposa canvis que afecten als requeriments inicials del projecte
Estimació	Temps de desenvolupament molt ajustat	El temps requerit per desenvolupar el nostre software està molt ajustat

Taula 6. Identificació de risc del projecte

[3] És una eina de treball en grup que facilita el sorgiment de noves idees sobre un tema o problema determinat. Es pot trobar àmplia informació sobre aquest tema a Internet. Citat de http://www.wikipedia.org

6.2 – ANÀLISI DE RISCS

En aquesta etapa, es considera cada risc identificat per separat i es decideix sobre la seva probabilitat i serietat. No hi ha una manera fàcil fer-ho, és de vital importància l'experiència del gestor del projecte.

La **probabilitat** del risc la podem valorar a partir dels següents intervals:

- **Molt baixa**: menor al 10 %
- **Baixa**: entre el 10% i el 25%
- **Moderada**: entre el 25% i el 50%
- **Alta**: entre el 50% i el 75%
- **Molt alta**: major al 75%

Els **efectes** del risc es poden valorar com:

- **Catastròfic**
- **Seriós**
- **Tolerable**
- **Insignificant**

A continuació podem veure l'anàlisi dels riscs del nostre projecte:

Risc	Probabilitat	Efecte
Desconeixement de les eines de treball	Moderada	Tolerable
Canvis en els requeriments	Moderada	Seriós
Temps de desenvolupament molt ajustat	Alta	Seriós

Taula 7. Anàlisi de riscs del projecte

S'ha de tenir en compte que tant la probabilitat com l'efecte del risc, canvien a mesura que es disposa de major informació sobre el risc, i els plans de gestió del mateix s'implementen. Per tant aquesta taula s'ha d'anar actualitzant a cada iteració del procés de riscs.

6.3 – PLANIFICACIÓ DE RISCS

El procés de planificació de riscs, considera cada un dels riscs identificats així com les estratègies per gestionar-los. Tampoc existeix un procés senzill que ens permeti establir els plans de gestió de riscs, depenent altra vegada de l'experiència del gestor del projecte.

Les estratègies per planificar els riscs es poden dividir en tres categories:

1. **Estratègies de prevenció:** ens serviran per reduir la probabilitat que el risc aparegui.
2. **Estratègies de minimització:** seguint aquestes estratègies es reduirà el impacte del risc.
3. **Plans de contingència:** significa estar preparat pel pitjor i tenir una estratègia per cada cas.

Seguidament mostrem la planificació dels riscs en el nostre projecte:

Risc	Categoria	Estratègia
Desconeixement de les eines de treball	Prevenció	Lectura de manuals i cursos de Microsoft Visual C# 2010, amb la conseqüent realització de proves pràctiques per la correcta comprensió dels continguts. Prestarem especial atenció als apartats més interessants de cara al nostre projecte.
	Minimització	A mesura que anem avançant amb el projecte, seguirem consultant manuals, cursos, ajudes i fòrums on-line per tal de resoldre els dubtes que vagin sorgint.
	Contingència	En el cas d'arribar en un punt d'estancament, i si no es pot dur a terme d'una altra manera, es consultarà a un professional extern.
Canvis en els requeriments	Prevenció	Efectuar les reunions necessàries amb el client per tal d'especificar els requeriments el màxim de precisos possible. Conscienciar al client en cada reunió del que significaria haver de canviar els requeriments una vegada començat el projecte.
	Minimització	A cada versió (iteració) del projecte, revisar de nou els requeriments i si fa falta reunir-nos amb el client.
	Contingència	Analitzar amb lupa els nous requeriments i tots els possibles efectes relacionats. Una vegada estudiat l'abast dels canvis, reunir-nos amb el client per revisar i acordar de nou la planificació i entrega del projecte.
Temps de desenvolupament molt ajustat	Prevenció	Mitjançant l'apartat de planificació del projecte, intentarem predir una data d'entrega òptima, tenint en compte possibles contratemps.
	Minimització	Després de cada part important del desenvolupament, prestar atenció en si anem complint amb la planificació del projecte. En cas de desviar-nos negativament, realitzar petits ajustos en la planificació.
	Contingència	Tant bon punt vegem que no podrem complir amb la data d'entrega prevista, acordarem una reunió amb el client per exposar-li els inconvenients apareguts i proposar-li una nova data d'entrega.

Taula 8. Estratègies de gestió dels riscs del projecte

6.4 – SUPERVISIÓ DE RISCS

La supervisió de riscs, normalment valora cadascun dels riscs identificats per decidir si aquests són més o menys probables i si han canviat els seus efectes. Aquests fets no els podem observar de forma directa, pel que buscarem altres factors que ens donin indicis de la probabilitat del risc i dels seus efectes. Aquests factors dependran dels tipus de risc.

La supervisió dels riscs ha de ser un procés continu. En cada revisió del progrés de gestió, cada un dels riscs clau ha de ser considerat i analitzat per separat.

En la següent taula, mostrem alguns dels possibles factors de risc en el nostre software:

Tipus	Risc	Factors
Personal	Desconeixement de les eines de treball	Alta complexitat en l'assimilació de conceptes, poca informació sobre les eines, excés de temps en apartats no útils per al projecte,...
Requeriments	Canvis en els requeriments	Moltes peticions de canvis, queixes del client, client amb les idees poc clares,...
Estimació	Temps de desenvolupament molt ajustat	Fracàs en la planificació del projecte, malalties del desenvolupador, incompliment dels proveïdors de hardware,...

Taula 9. Factors de risc del projecte

7 – PLANIFICACIÓ I PRESSUPOST

En aquest capítol detallarem la planificació que es seguirà durant aquest projecte, les tasques a desenvolupar i el temps estimat per cada una d'elles. Per altra banda, també mostrarem el cost que tindria aquest software en el cas que s'arribés a comercialitzar.

7.1 – PLANIFICACIÓ

La gestió d'un projecte de software comença amb un conjunt d'activitats que s'anomenen *planificació del projecte*. Abans de que el projecte comenci, el gestor i l'equip de software han de dur a terme una estimació del treball a realitzar, dels recursos necessaris i del temps que passarà des del inici fins a l'entrega del projecte. Sempre que fem estimacions, estem mirant al futur i acceptem cert grau d'incertesa. Per aquest fet, és importantíssim l'experiència anterior de totes les persones involucrades en el projecte. [PRE02]

A continuació, mostrem una taula amb el detall de totes les tasques del nostre projecte ordenades cronològicament, així com la duració estimada de totes elles. S'assumeix que una jornada laboral té una durada de 8 hores. Tots els períodes indicats inclouen tant la data d'inici com la data de fi especificades. No s'ha treballat en el projecte cap dia festiu ni caps de setmana.

Tasca	Duració (dies)	Data inici	Data fi
1. Obtenció de requeriments	**51**	**09/05/12**	**19/07/12**
1.1. Reunions amb el client	51[4]	09/05/12	19/07/12
1.2. Anàlisi de requeriments	51[5]	09/05/12	19/07/12
2. Planificació de tasques	**2**	**20/07/12**	**23/07/12**
3. Base de dades	**24**	**24/07/12**	**27/08/12**
3.1. Model entitat-relació	11	24/07/12	07/08/12
3.2. Model relacional	8	08/08/12	20/08/12
3.3. Diccionari de dades	5	21/08/12	27/08/12
4. Disseny de l'aplicació	**37**	**28/08/12**	**19/10/12**
4.1. Disseny capa d'accés a dades	12	28/08/12	13/09/12
4.2. Disseny capa de negoci	11	14/09/12	28/09/12
4.3. Disseny capa d'interfície	14	01/10/12	19/10/12
5. Implementació	**109**	**22/10/12**	**28/03/13**
5.1. Aprenentatge de l'eina	109[6]	22/10/12	28/03/13

[4] Hem indicat el total de dies en l'obtenció de requeriments, tant en l'apartat de reunions amb el client com en l'anàlisi de requeriments. Això es degut a que l'anàlisi de requeriments conviurà i es nodrirà de les reunions setmanals efectuades amb el client.

[5] Tenint en compte que per al desenvolupament del software hem escollit el model incremental, l'obtenció de requeriments no estarà situada únicament al inici de la planificació, sinó que durant el desenvolupament també poden sorgir reunions i anàlisi de nous requeriments. Per tant, en aquesta tasca només hem quantificat l'obtenció dels requeriments inicials.

5.2. Creació de la Base de Dades	4	22/10/12	25/10/12
5.3. Capa d'accés a dades	8	26/10/12	07/11/12
5.4. Capa de negoci			
5.4.1. Classe per consultar o guardar dades a través de la capa d'accés a dades	11	08/11/12	22/11/12
5.4.2. Classe per les còpies de seguretat	2	23/11/12	26/11/12
5.4.3. Classe per generar de codis de barres	2	27/11/12	28/11/12
5.5. Capa d'interfície			
5.5.1. Controls específics i bàsics per la nostra aplicació (botonera, buscador,...).	9	29/11/12	12/12/12
5.5.2. Formularis			
5.5.2.1. Formulari base per manteniments	10	13/12/12	28/12/12
5.5.2.2. Manteniment d'articles	5	31/12/12	07/01/13
5.5.2.3. Manteniment de famílies	1	08/01/13	08/01/13
5.5.2.4. Manteniment de proveïdors	2	09/01/13	10/01/13
5.5.2.5. Manteniment de clients	2	11/01/13	14/01/13
5.5.2.6. Manteniment de venedors	1	15/01/13	15/01/13
5.5.2.7. Manteniment de caixes TPV	2	16/01/13	17/01/13
5.5.2.8. Manteniment d'usuaris	2	18/01/13	21/01/13
5.5.2.9. Formulari de cerques	3	22/01/13	24/01/13
5.5.2.10. Formulari base per documents	12	25/01/13	11/02/13
5.5.2.11. Document de compra	3	12/02/13	14/02/13
5.5.2.12. Document de venda (T.P.V.)	9	15/02/13	27/02/13
5.5.2.13. Formulari base per impressions	3	28/02/13	04/03/13
5.5.2.14. Llistat de vendes	2	05/03/13	06/03/13
5.5.2.15. Impressió d'etiquetes EAN	1	07/03/13	07/03/13
5.5.2.16. Gestió de còpies de seguretat	4	08/03/13	13/03/13
5.5.2.17. Formulari d'entrada/identificació	1	14/03/13	14/03/13
5.5.2.18. Formulari principal	2	15/03/13	18/03/13
5.5.3. Informes / reports			
5.5.3.1. Document de venda	1	19/03/13	19/03/13
5.5.3.2. Llistat de vendes	2	20/03/13	21/03/13
5.5.3.3. Etiquetes EAN	2	22/03/13	25/03/13
5.5.4. Icones i imatges	3	26/03/13	28/03/13
6. Avaluació i proves	**19**	**02/04/13**	**26/04/13**
7. Documentació	**29**[7]	**29/04/13**	**07/06/13**
8. Totalització de la planificació	**271**	**09/05/12**	**07/06/13**

Taula 10. Planificació del projecte

[6] En la tasca d'aprenentatge de l'eina, he indicat el total de dies de la implementació, tenint en compte que m'he estat formant durant tot el procés de desenvolupament de l'aplicació.

[7] S'ha establert el període per la documentació del projecte en l'última tasca de la planificació, no perquè es realitzi tota la documentació al final del projecte, sinó per dur a terme una revisió exhaustiva de la documentació feta durant tot el projecte.

7.2 – PRESSUPOST

En aquest apartat veurem el detall de totes les despeses efectuades en la realització d'aquest projecte, tot agrupant-ho per diferents conceptes.

Podem trobar els següents grups de despeses:

- **Personal:** contempla el cost de totes les hores dedicades al projecte per part del personal participant. Considerant que aquest projecte s'ha dut a terme com a treball final de carrera, i que hi havia un desconeixement total del llenguatge de programació a utilitzar, hem valorat el cost d'una hora de treball en 10€.

  El càlcul total s'obté mitjançant la següent fórmula:

  Cost(€) = Hores dedicades * Cost hora

- **Inventariable:** aquest grup inclourà l'amortització dels equips informàtics comprats específicament per al desenvolupament del projecte. El període d'amortització per equips informàtics, es considera de tres anys.

  L'amortització es calcula de la manera següent:

  Cost(€) = (A/B) * C * D

  On: A = Número de mesos que s'ha utilitzat l'equip en el projecte
  B = Període d'amortització (36 mesos per equips informàtics)
  C = Cost de l'equip
  D = Percentatge d'ús de l'equip en el projecte

- **Fungibles:** són tots aquells bens que s'han consumit o esgotat durant la consecució del projecte, com per exemple: papers, bolígrafs, tintes impressora,...

- **Dietes i viatges:** es tindrà en compte el cost de les dietes i viatges realitzats exclusivament pel projecte. Per al càlcul d'aquests costs, ens fixarem en el reial decret 439/2007 del reglament del IRPF, que estableix un cost de 0.19 euros per quilòmetre i 26.67 euros per dia en concepte de dieta.

- **Costs indirectes:** aquests costs també anomenats "Overheads", representen totes aquelles despeses que no es poden repercutir directament al projecte, però que estan lligades a activitats del projecte. Per aquest grup hem arribat a un acord amb el client, per tal que aquestes despeses suposin el 10% sobre el total del pressupost.

- **Llicències de software:** en aquest apartat hi detallarem els costs de les llicències dels diferents softwares usats en l'elaboració del projecte.

Tot seguit presentem el detall de totes les despeses associades a l'elaboració d'aquest projecte mitjançant aquest pressupost en forma de taula.

Concepte	Quantitat	Preu (€)	Import (€)
1. Personal			
▪ Hores implementació (109 dies)	872	10,00	8.720,00
▪ Hores avaluació i proves (19 dies)	152	10,00	1.520,00
Total Personal			**10.240,00**
2. Inventariable			
▪ PC Intel i7	1	604,00	
▪ Monitor 22"	1	115,00	
▪ Teclat + ratolí	1	33,00	
▪ Impressora làser	1	57,00	
▪ Disc dur extern 1TB	1	69,00	
▪ Lector de codis de barres	1	76,00	
Amortització = (13/36) * 954 * (100/100) = 344,50		954,00	
Total Iventariable			**344,50**[8]
3. Fungible			
▪ Paquet de 500 folis DIN A4	1	4,00	4,00
▪ Pack 4 bolígrafs PILOT	2	8,00	16,00
▪ Tòner impressora	1	18,00	18,00
▪ Pack 25 DVD	1	10,00	10,00
Total Fungible			**48,00**
4. Dietes i viatges			
▪ Dietes (4 reunions de dia sencer)	4	26,66	106,64
▪ Viatges (12 reunions)	540	0,19	102,60
Total Dietes i viatges			**209,24**
5. Llicències de software			
▪ Microsoft Windows 7 Ultimate	1	188,00	188,00
▪ Microsoft Office Professional 2007	1	337,00	337,00
▪ Microsoft Visual Studio 2010 Professional	1	937,00	937,00
▪ SAP Crystal Reports	1	297,00	297,00
Total Llicències			**1.759,00**
6. SUBTOTAL			
SUBTOTAL			**12.600,74**
7. Costs indirectes (overhead) 10%			
Total Overhead			**1.260,07**
8. TOTAL PRESSUPOST			
TOTAL PRESSUPOST			**13.860,81€**

Taula 11. Pressupost del projecte

Pressupost valorat sense IVA, caldrà afegir-hi el 21% d'IVA reglamentari.

[8] Càlcul realitzat mitjançant la fórmula d'amortització indicada en el grup de despeses d'inventariable.

8 – BASE DE DADES

Una base de dades és un conjunt de dades que pertanyen a un mateix context, i s'emmagatzemen sistemàticament per al seu posterior ús. Una de les formes més comuns per classificar les bases de dades, és d'acord al seu model d'administració de dades. Segons aquesta classificació, les bases de dades relacionals són les més utilitzades actualment per modelar problemes reals i administrar les dades dinàmicament.

El disseny d'una base de dades consisteix en definir l'estructura de les dades que ha de contenir un sistema determinat. En el cas relacional, aquesta estructura serà un conjunt d'esquemes de relació, amb els seus atributs, claus primàries, claus foranes,...

Degut a la complexitat del disseny de la base de dades, es divideix en tres etapes:

1. **Etapa del disseny conceptual:** en aquesta etapa s'obté una estructura de la informació que haurà d'emmagatzemar la futura base de dades, independentment de la tecnologia que s'utilitzi. El resultat d'aquesta etapa s'expressa mitjançant algun model de dades d'alt nivell, un dels més utilitzats és el model entitat-relació.

2. **Etapa del disseny lògic:** a partir del resultat del disseny conceptual, aplicarem un seguit de transformacions per apropar-lo a la tecnologia escollida. Si es tracta d'un sistema de gestió de bases de dades relacional, obtindrem un conjunt de relacions amb els seus atributs, claus primàries i claus foranes(model relacional).

3. **Etapa del disseny físic:** consisteix en transformar l'estructura obtinguda en el disseny lògic amb l'objectiu d'aconseguir una major eficiència. Es tindran en compte aspectes de la implementació física com l'elecció d'estructures, mides i característiques dels processos que consulten i actualitzen la base de dades.

8.1 – MODEL ENTITAT-RELACIÓ

Es tracta del model conceptual per excel·lència. És el model que hem utilitzat per al modelatge de la base de dades del nostre projecte, degut a la seva simplicitat i llegibilitat. Està basat en una percepció del món real, consistent en objectes bàsics anomenats **entitats** i en les **relacions** entre aquests objectes.

Una **entitat** és un objecte del món real sobre el que volem emmagatzemar informació. Les entitats estan compostes per atributs, que són les dades que defineixen l'objecte. Mitjançant un rectangle amb el nom de l'entitat al seu interior, és com representem les entitats gràficament dins del model entitat-relació. També hi ha unes altres entitats anomenades **febles**, que depenen d'una altra entitat per poder existir. Aquestes les representem igual que la resta d'entitats però amb un doble rectangle.

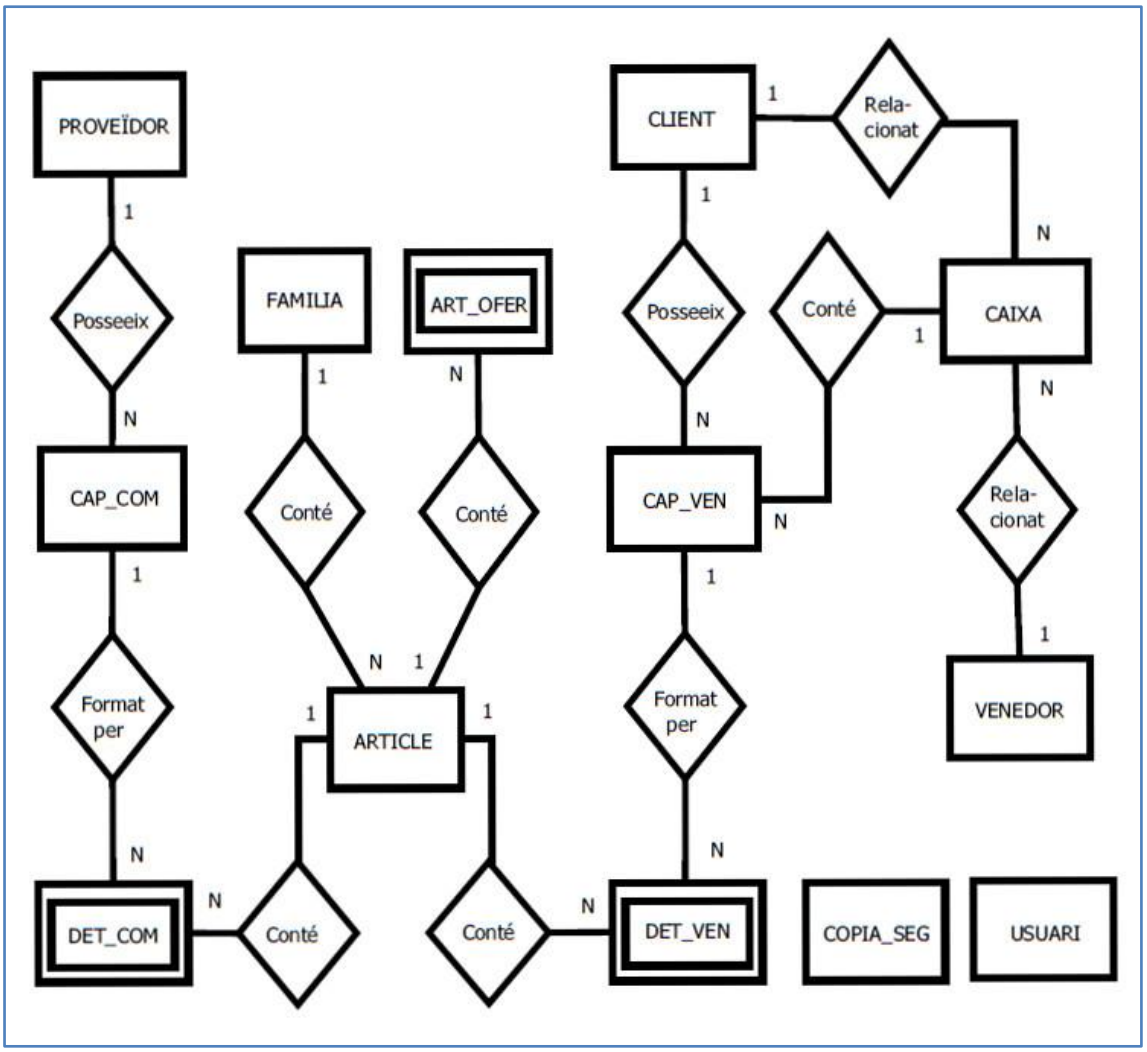
Anomenem **relació** a les correspondències o associacions entre diferents entitats. Les relacions entre entitats es representen gràficament amb rombes.

Donat un conjunt de relacions binaries i els conjunts d'entitats A i B podem tenir els següents tipus de relacions:

- **1 a 1:** una entitat de A es relaciona únicament amb una entitat de B i viceversa.
- **1 a N:** una entitat de A es relaciona amb cap o moltes entitats de B, però una entitat de B es relaciona amb una única entitat de A.
- **N a 1:** una entitat de A es relaciona exclusivament amb una entitat de B, però una entitat de B es pot relacionar amb cap o moltes entitats de A.
- **N a N:** una entitat de A es pot relacionar amb cap o moltes entitats de B i viceversa.

Els **atributs** són característiques d'interès sobre una entitat o relació, i en representen les seves propietats bàsiques.

Per la implementació del model entitat-relació, ens hem servit d'un programari de lliure distribució anomenat DIA[9], que ens ha permès dissenyar-lo d'una forma ràpida i intuïtiva.



Il·lustració 9. Model entitat-relació del projecte

[9] Software obtingut a través de https://live.gnome.org/Dia, on a més de la descàrrega per diferents plataformes, també hi podem trobar força documentació com manuals, guies, exemples, diversos enllaços d'interès,...

8.2 – MODEL RELACIONAL

Es tracta del model de disseny lògic de bases de dades més estès i utilitzat pels dissenyadors en l'actualitat.

En el model relacional definirem totes les entitats que apareixen en el model entitat-relació obtingut en l'apartat anterior, tot detallant els seus atributs, claus primàries i claus foranes.

Tot seguit passem a definir el model relacional utilitzat per dissenyar la base de dades de la nostra aplicació.

- **ARTICLE** (codi, nom, familia, proveidor, tpcmarge, drevisio, ultcost, foto, observa)
- **ART_OFER** (article, linia, dataini, datafi, preu, dte)
- **FAMILIA** (codi, nom, tpcmarge)
- **PROVEIDOR** (codi, nom, cif, telefon, poblacio, direccio, cpostal, provincia, email, web, contacte, observa, dte1, dte2)
- **CLIENT** (codi, nom, cif, telefon, poblacio, direccio, cpostal, provincia, observa)
- **VENEDOR** (codi, nom, observa)
- **CAIXA** (codi, nom, clicomptat, venedor, observa)
- **CAP_COM** (codi, numero, proveidor, data, observa)
- **DET_COM** (codi, linia, article, unitats, preu, dte1, dte2, import)
- **CAP_VEN** (codi, numero, caixa, venedor, client, data, observa)
- **DET_VEN** (codi, linia, article, unitats, preu, dte, import)
- **USUARI** (nom, password, tipus)
- **COPIA_SEG** (codi, data, fitxer)

8.3 – DICCIONARI DE DADES

Tot seguit detallarem l'estructura de totes les taules que formen part de la base de dades del nostre projecte. Indicarem el nom de la taula, el nom dels camps que la formen, el tipus de dades de cada camp i la seva longitud. També indicarem quins camps formen part de la clau primària i de la clau forana.

Per la definició del diccionari de dades utilitzarem la nomenclatura dels tipus de dades propis del gestor de bases de dades SQL Server, ja que aquest és el gestor de base de dades escollit per gestionar i emmagatzemar les dades en la nostra aplicació.

ARTICLE

Nom del camp	Tipus de dades	Longitud	Clau primària	Clau forana
codi	nvarchar	10	SI	NO
nom	nvarchar	50	NO	NO
familia	nchar	3	NO	SI
proveidor	nchar	5	NO	SI
tpcmarge	numeric	5,2	NO	NO
drevisio	datetime		NO	NO
ultcost	numeric	15,2	NO	NO
foto	nvarchar	200	NO	NO
observa	text		NO	NO

Taula 12. Taula Article

FAMILIA

Nom del camp	Tipus de dades	Longitud	Clau primària	Clau forana
codi	nchar	3	SI	NO
nom	nvarchar	50	NO	NO
tpcmarge	numeric	5,2	NO	NO

Taula 13. Taula Família

PROVEIDOR

Nom del camp	Tipus de dades	Longitud	Clau primària	Clau forana
codi	nchar	5	SI	NO
nom	nvarchar	50	NO	NO
cif	nchar	10	NO	NO
telefon	nvarchar	20	NO	NO
poblacio	nvarchar	50	NO	NO
direccio	nvarchar	100	NO	NO
cpostal	nchar	10	NO	NO
provincia	nvarchar	50	NO	NO
email	nvarchar	200	NO	NO
web	nvarchar	200	NO	NO
contacte	nvarchar	50	NO	NO
observa	text		NO	NO
dte1	numeric	5,2	NO	NO
dte2	numeric	5,2	NO	NO

Taula 14. Taula Proveïdor

CLIENT

Nom del camp	Tipus de dades	Longitud	Clau primària	Clau forana
codi	nchar	5	SI	NO
nom	nvarchar	50	NO	NO
cif	nchar	10	NO	NO
telefon	nvarchar	20	NO	NO
poblacio	nvarchar	50	NO	NO
direccio	nvarchar	100	NO	NO
cpostal	nchar	10	NO	NO
provincia	nvarchar	50	NO	NO
email	nvarchar	200	NO	NO
web	nvarchar	200	NO	NO
observa	text		NO	NO

Taula 15. Taula Client

VENEDOR

Nom del camp	Tipus de dades	Longitud	Clau primària	Clau forana
codi	nchar	3	SI	NO
nom	nvarchar	50	NO	NO
observa	text		NO	NO

Taula 16. Taula Venedor

CAIXA

Nom del camp	Tipus de dades	Longitud	Clau primària	Clau forana
codi	nchar	3	SI	NO
nom	nvarchar	50	NO	NO
clicomptat	nchar	5	NO	SI
venedor	nchar	3	NO	SI
observa	text		NO	NO

Taula 17. Taula Caixa

CAP_COM

Nom del camp	Tipus de dades	Longitud	Clau primària	Clau forana
codi	nchar	36	SI	NO
numero	nvarchar	10	NO	NO
proveidor	nchar	5	NO	SI
data	datetime		NO	NO
observa	text		NO	NO

Taula 18. Taula Cap_Com

DET_COM

Nom del camp	Tipus de dades	Longitud	Clau primària	Clau forana
codi	nchar	36	SI	NO
linia	int		SI	NO
article	nvarchar	10	NO	SI
unitats	numeric	20,2	NO	NO
preu	numeric	20,2	NO	NO
dte1	numeric	5,2	NO	NO
dte2	numeric	5,2	NO	NO
import	numeric	20,2	NO	NO

Taula 19. Taula Det_Com

CAP_VEN

Nom del camp	Tipus de dades	Longitud	Clau primària	Clau forana
codi	nchar	36	SI	NO
numero	nvarchar	10	NO	NO
caixa	nchar	3	NO	SI
venedor	nchar	3	NO	SI
client	nchar	5	NO	SI
data	datetime		NO	NO
observa	text		NO	NO

Taula 20. Taula Cap_Ven

DET_VEN

Nom del camp	Tipus de dades	Longitud	Clau primària	Clau forana
codi	nchar	36	SI	NO
linia	int		SI	NO
article	nvarchar	10	NO	SI
unitats	numeric	20,2	NO	NO
preu	numeric	20,2	NO	NO
dte	numeric	5,2	NO	NO
import	numeric	20,2	NO	NO

Taula 21. Taula Det_Ven

USUARI

Nom del camp	Tipus de dades	Longitud	Clau primària	Clau forana
nom	nvarchar	20	SI	NO
password	nvarchar	20	NO	NO
tipus	tinyint	1	NO	NO

Taula 22. Taula Usuari

COPIA_SEG

Nom del camp	Tipus de dades	Longitud	Clau primària	Clau forana
codi	nchar	36	SI	NO
data	datetime		NO	NO
fitxer	text		NO	NO

Taula 23. Taula Copia_Seg

9 – DISSENY DE L'APLICACIÓ

En aquest apartat l'objectiu consistirà en documentar les diferents funcionalitats/tasques de la nostra aplicació de forma estructural. Per dur a terme aquest anàlisi de tasques hi ha diverses tècniques, però la més utilitzada és l'Anàlisi Jeràrquic de Tasques que presentem a continuació.

9.1 – ANÀLISI JERÀRQUIC DE TASQUES (HTA)

L'anàlisi Jeràrquic de Tasques, traduït de l'acrònim anglès HTA (*Hierarchical Task Analysis*), és la tècnica d'anàlisi de tasques més coneguda i utilitzada. En aquesta tècnica es mostra una descripció de les tasques en termes d'**operacions** i **plans**.

Operacions: són les activitats que realitzen les persones per assolir un objectiu. Les operacions es poden descompondre de forma jeràrquica i s'assigna un pla a cada una de les subtasques que apareixen.

Plans: són una descripció de les condicions que s'han de donar quan es porta a cap cadascuna de les activitats.

Objectiu: es defineix com un estat determinat del sistema que pot/vol aconseguir l'usuari.

El format gràfic mitjançant el que representarem cada tasca, és molt similar al d'un arbre amb branques i subbranques en funció de les necessitats [SHE89].

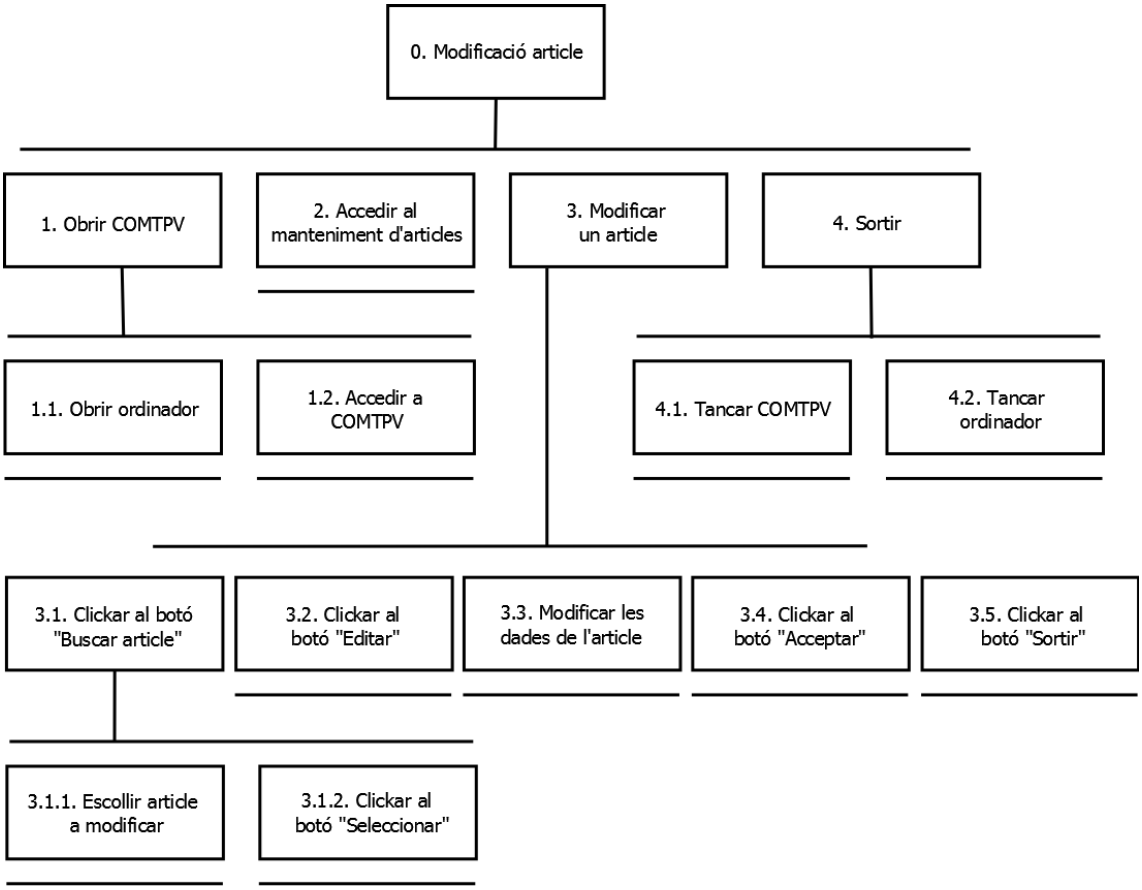
En la nostra aplicació, les tasques més importants són les següents:

- Alta d'article / família / client / proveïdor / venedor / caixa TPV / usuari
- Baixa d'article / família / client / proveïdor / venedor / caixa TPV / usuari
- Modificació d'article / família / client / proveïdor / venedor / caixa TPV / usuari
- Compra d'articles
- Venda d'articles
- Llistat de vendes
- Impressió d'etiquetes d'articles (codis de barres)
- Realitzar còpia de seguretat
- Restaurar còpia de seguretat

Tot seguit mostrarem l'anàlisi jeràrquic d'algunes d'aquestes tasques més importants, la resta de tasques les podrem veure en l'annex de l'aprartat 16.2.

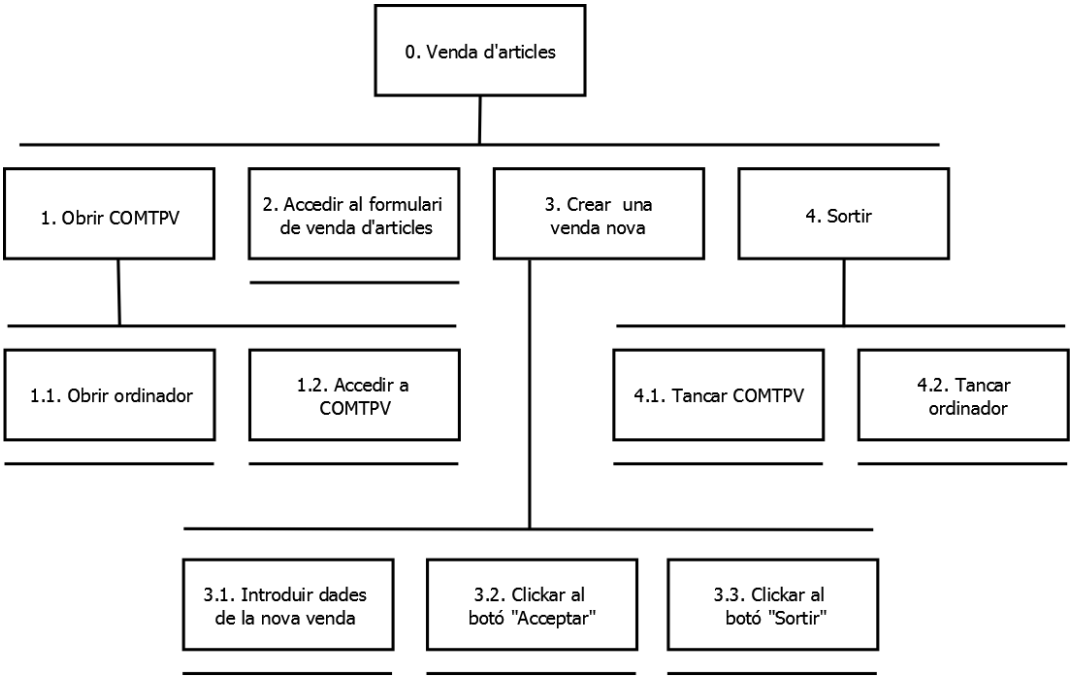
Tasca 1. Modificació d'article / família / client / proveïdor / venedor / caixa TPV / usuari

Hem englobat en aquesta tasca totes les tasques de modificació del nostre software, donat que la operativa és exactament la mateixa.



Il·lustració 10. Tasca "Modificació article"

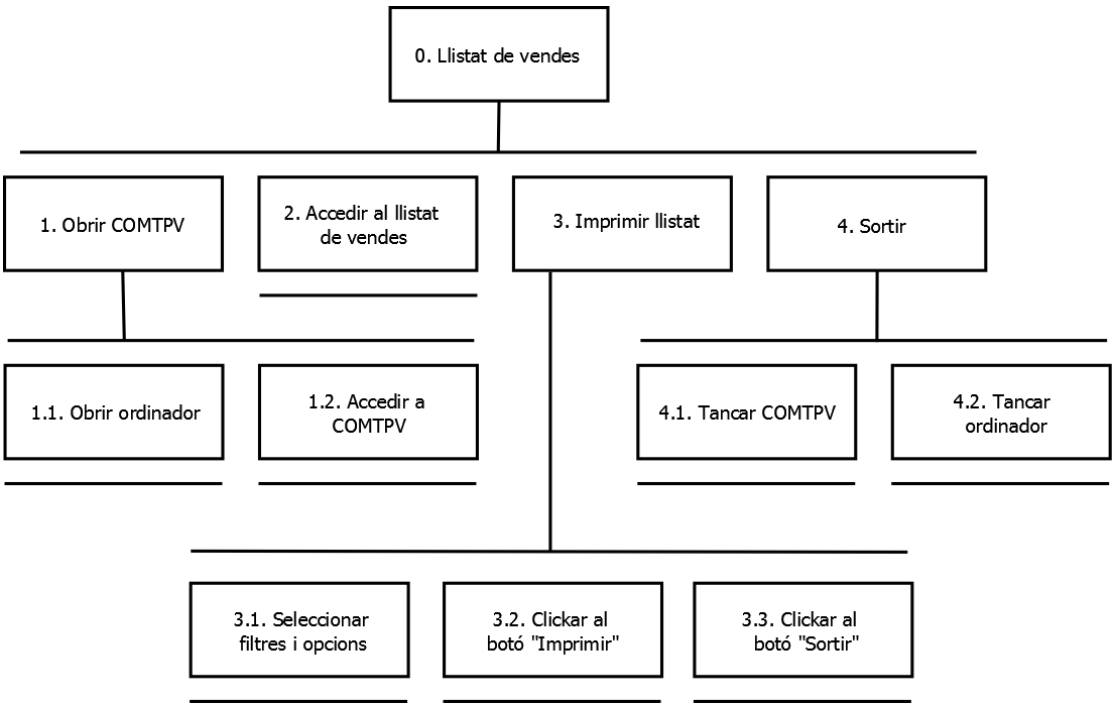
Tasca 2. Venda d'articles



Il·lustració 11. Tasca "Venda d'articles"

Cal comentar que cabria una segona tasca per la "Venda d'articles" corresponent als usuaris de tipus "Venedor". En l'anàlisi d'aquesta nova tasca, s'eliminaria la subtasca 2 ja que al obrir COMTPV, aniríem directament al formulari per la venda d'articles.

Tasca 3. Llistat de vendes



Il·lustració 12. Tasca "Llistat de vendes"

9.2 – DISSENY DE LA INTERFÍCIE

En aquest apartat explicarem les diferents decisions de disseny que s'han pres durant el desenvolupament.

Un dels punts que hem tingut molt present, és la possible poca experiència dels usuaris finals en el maneig d'aplicacions de software. Per aquest fet s'ha intentat dissenyar una interfície el més intuïtiva i fàcil d'usar. També s'ha seguit el mateix patró per dissenyar formularis del mateix tipus, d'aquesta manera l'usuari sempre sabrà on està tot situat i quina és la seva funció. Amb tot això queda clar que la usabilitat juga un paper importantíssim en el disseny de la interfície. Usabilitat es defineix col·loquialment com a facilitat d'ús, ja sigui referent a una pàgina web, una aplicació informàtica o qualsevol altre sistema que interactuï amb un usuari.

Cal anomenar que en tots els formularis, s'ha tingut en compte que l'usuari pugui realitzar les accions i operacions més importants a través del teclat, sense haver d'usar el ratolí i per tant poder treballar de forma molt ràpida. Al prémer la tecla "Alt", es mostra en diferents objectes del formulari un subratllat en les lletres d'etiquetes, botons,... indicant quina és la combinació de tecles a pitjar per dur a terme cada acció.

La nostra aplicació està formada per un formulari principal, amb un menú lateral de grans icones a la part esquerra. Mitjançant aquestes distintives icones, podrem accedir a tots els formularis/opcions disponibles. Cal tenir en compte que el formulari principal amb el seu menú lateral sempre estarà visible per l'usuari, i així poder accedir a qualsevol opció en tot moment. Tot formulari que anem obrint quedarà encaixat dins del formulari principal i conviurà amb altres formularis que haguem obert, per tant no treballarem de forma modal[10] i li donarem llibertat a l'usuari.

Acte seguit fem una enumeració de tots els formularis presents en l'aplicació:

- Formulari d'autenticació
- Formulari principal
- Formulari de cerques
- Manteniment d'articles
- Manteniment de famílies
- Manteniment de proveïdors
- Manteniment de clients
- Manteniment de venedors
- Manteniment de caixes T.P.V.
- Manteniment d'usuaris
- Document de compra
- Document de venda (T.P.V.)
- Llistat de vendes
- Impressió d'etiquetes d'articles
- Gestió de còpies de seguretat

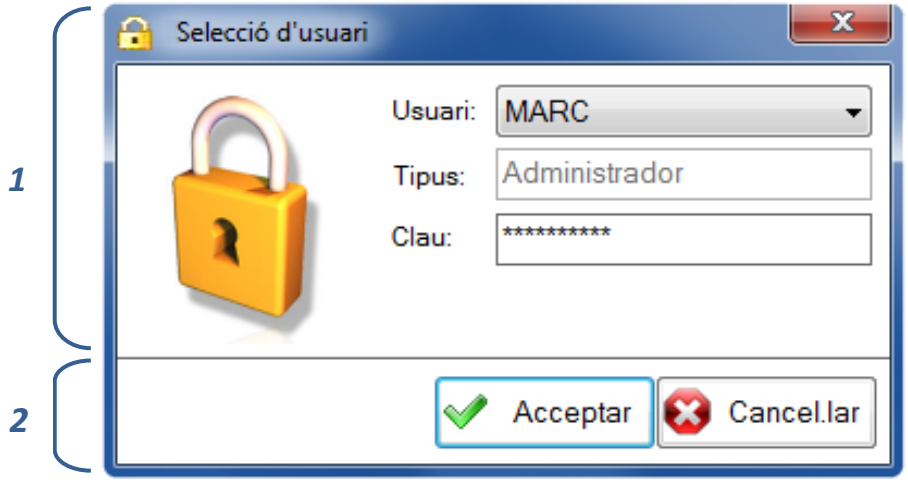
[10] Un formulari modal és aquell que obliga a l'usuari a tancar-lo per tal de poder continuar treballant amb la resta de l'aplicació.

9.3 – ESTRUCTURACIÓ DELS FORMULARIS

Utilitzant captures d'imatge del nostre software, mostrarem l'estructura dels diferents tipus de formularis que hi podem trobar, tot explicant les parts de que es composen.

9.3.1 – FORMULARI D'AUTENTICACIÓ

El formulari d'autenticació és el primer formulari que veurà l'usuari al entrar en l'aplicació. Aquest formulari tindrà dos propòsits, el primer i més evident serà el d'autenticar l'usuari dins la nostra aplicació, i el segon consistirà en obtenir el tipus d'usuari (administrador o venedor), que ens servirà per habilitar les opcions corresponents al tipus del usuari autenticat.



Il·lustració 13. Formulari d'autenticació

Com podem observar en la il·lustració 13, el formulari d'autenticació està dividit en dos parts:

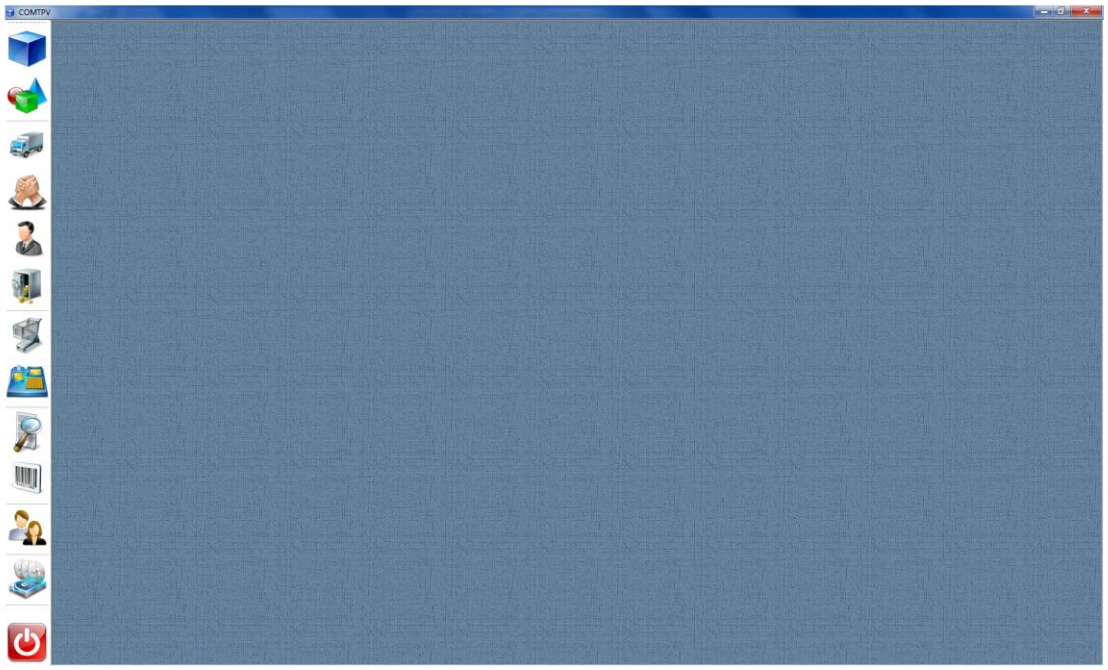
1. En la part superior podrem escollir l'usuari mitjançant un quadre combinat (*combobox)*[11], on hi veurem tots els usuaris disponibles. Segons l'usuari escollit es mostrarà el seu tipus corresponent. També hi haurem d'indicar la clau d'accés per a l'usuari escollit
2. En la part inferior només hi disposarem dels botons d'acceptar i cancel·lar.

9.3.2 – FORMULARI PRINCIPAL

El formulari principal amb el seu menú lateral, com ja havíem anomenat anteriorment, estarà actiu en tot moment per als usuaris de tipus "administrador" una vegada aquests s'hagin autenticat. D'aquesta manera podran accedir a totes les opcions disponibles en qualsevol

[11] És un control d'interfície gràfica d'usuari molt comú que consisteix en combinar en un mateix control les característiques d'un camp de text i d'una llista.

moment. Recordem que totes les opcions/formularis que anem obrint quedaran encaixades dins el formulari principal.



Il·lustració 14. Formulari principal

1 *2*

Tal com podem veure en la il·lustració 14, es diferencien dues zones en el formulari principal:

1. Menú lateral amb accés a totes les opcions de la nostra aplicació.
2. Formulari contenidor on s'aniran encaixant els formularis que anem obrint.

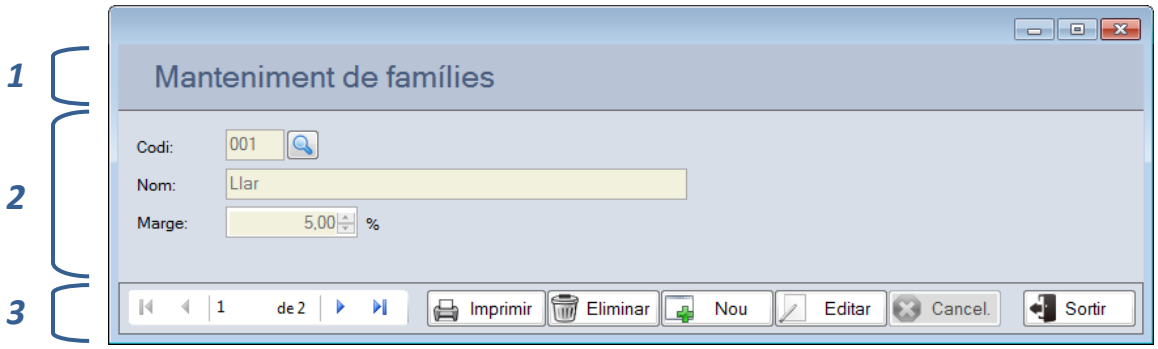
9.3.3 – FORMULARI DE MANTENIMENTS

En la nostra aplicació hi podem trobar els següents formularis de tipus "manteniment":

- Manteniment d'articles
- Manteniment de famílies
- Manteniment de proveïdors
- Manteniment de clients
- Manteniment de venedors
- Manteniment de caixes T.P.V.
- Manteniment d'usuaris

Tots aquests formularis parteixen de la mateixa base i estan estructurats d'igual manera. D'aquesta forma si en un futur volguéssim incorporar un nou manteniment a l'aplicació, ja tindríem gran part de la feina feta.

Hem escollit el formulari del manteniment de famílies per representar aquest tipus de formulari, ja que és un manteniment bastant bàsic en el qual hi podem veure la essència dels manteniments.



Il·lustració 15. Formulari de tipus "manteniment"

Els formularis de tipus “manteniment” estan estructurats en 3 parts:

1. **Títol**: hi ha el títol del manteniment.

2. **Cos**: conté les dades específiques del registre de la base de dades seleccionat. Cal destacar que tots els manteniments, com a mínim mostraran els camps codi i nom. També hi ha el botó amb icona de lupa(accessible mitjançant la tecla F3) per poder realitzar cerques més concretes d’un registre.

3. **Peu**: en aquest apartat hi trobarem les botoneres de navegació i d’accions.

   La **botonera de navegació** ens permetrà desplaçar-nos pels diferents registres del manteniment en qüestió a través d’aquests moviments:
   - Desplaçament al **primer** registre
   - Desplaçament al registre **anterior** a l’actual
   - Desplaçament al registre **següent** a l’actual
   - Desplaçament a **l’últim** registre

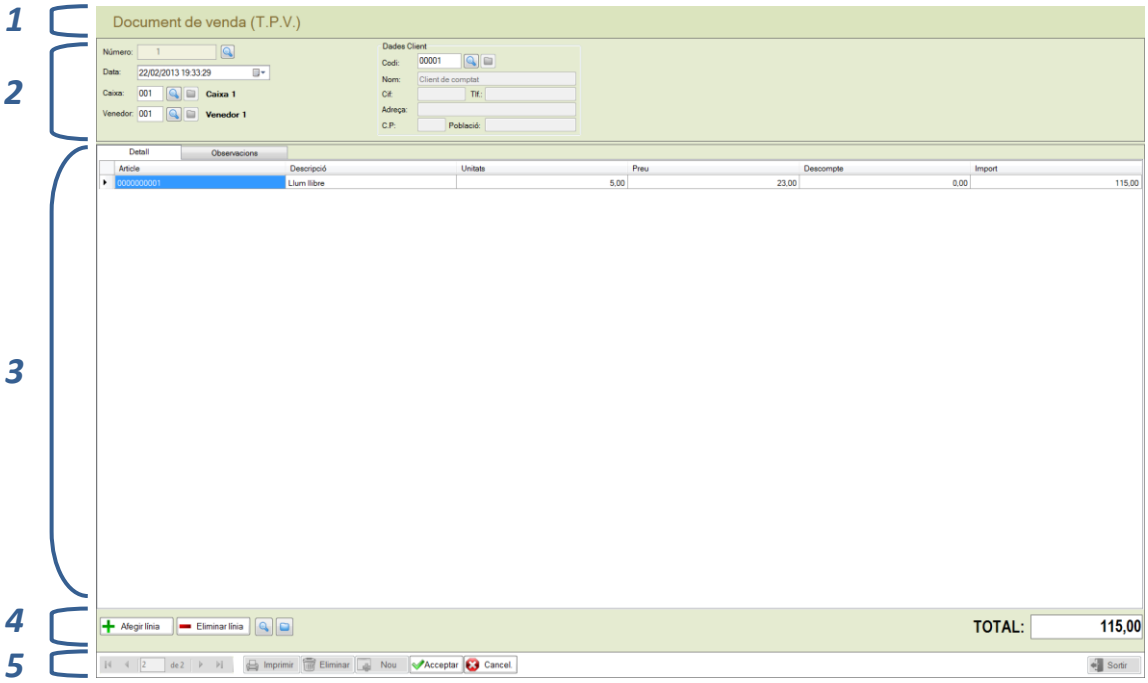
   Mitjançant la **botonera d’accions** realitzarem totes les operacions disponibles per als manteniments:

   - **Imprimir**: Imprimir una llista de tots els registres del manteniment actual.
   - **Eliminar**: Eliminar el registre actual de la base de dades.
   - **Nou**: Afegir un registre nou a la base de dades.
   - **Editar / Acceptar**: Botó amb doble utilitat, ens servirà per poder editar les dades del registre actual i posteriorment per acceptar els canvis realitzats.
   - **Cancel·lar**: Descarta els possibles canvis realitzats en l’edició o creació.
   - **Sortir**: Tancar el formulari de manteniment actiu.

9.3.4 – FORMULARI DE DOCUMENTS

En el nostre software només hi tenim dos formularis de tipus “document”, el document per realitzar les compres als nostres proveïdors i el document de venda. Podríem dir que aquest últim és el més rellevant de l’aplicació, degut a que tota la resta de l’aplicació gira al seu voltant.

Ambdós formularis s’han dissenyat a partir del mateix patró i per tant tenen la mateixa estructura. Cal anomenar també, que si en una ampliació futura es volgués afegir un nou document al nostre software, es partiria de la mateixa base.



Il·lustració 16. Formulari de tipus "document"

Podem distingir les parts següents en els formularis de tipus “document”:

1. **Títol:** títol del document en qüestió.

2. **Capçalera:** hi podem veure les dades del registre seleccionat referents a la capçalera del document. A través del botó amb icona de lupa(tecla F3) situat al costat del número del document, podrem cercar el document desitjat.

3. **Detall:** conté tots els registres de detall relacionats amb el document actual. A més a més, podem apreciar una segona pestanya d’observacions generals del document.

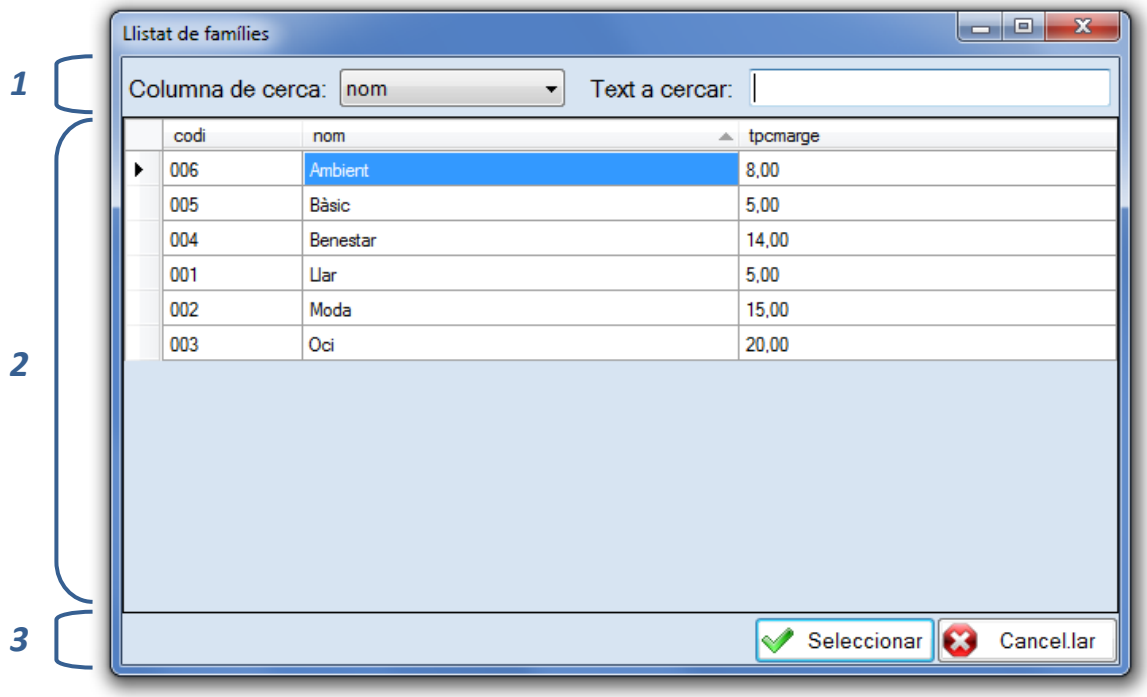
4. **Peu del detall:** hi podem trobar les operacions disponibles a realitzar en el detall. També hi ha la casella amb l’import total del document.

5. **Peu del document:** inclou les botoneres d'accions i de navegació tal com en els formularis de tipus "manteniment". La operativa d'aquestes botoneres és exactament la mateixa, per tant podeu consultar-la en l'apartat anterior.

9.3.5 – FORMULARI DE CERQUES

Cal comentar que el formulari de cerques és el mateix en totes les parts de programa. Dit d'una altra manera, ens serveix tant per realitzar cerques d'un registre de qualsevol manteniment com per buscar un document.

A continuació podem veure l'exemple de la cerca de famílies.



Il·lustració 17. Formulari de cerques

L'estructura del formulari de cerques és bastant senzilla, tot seguit la detallem:

1. **Capçalera:** conté un quadre combinat mitjançant el que podrem seleccionar la columna per la qual volem realitzar la cerca. Just al costat i tenim un quadre de text on a mesura que anem escrivint, ens anirà filtrant els registres coincidents.

2. **Detall:** consisteix en una graella on es mostren per defecte tots els registres de la taula de cerca actual, i a mesura que introduïm text a cercar aniran quedant els registres corresponents.

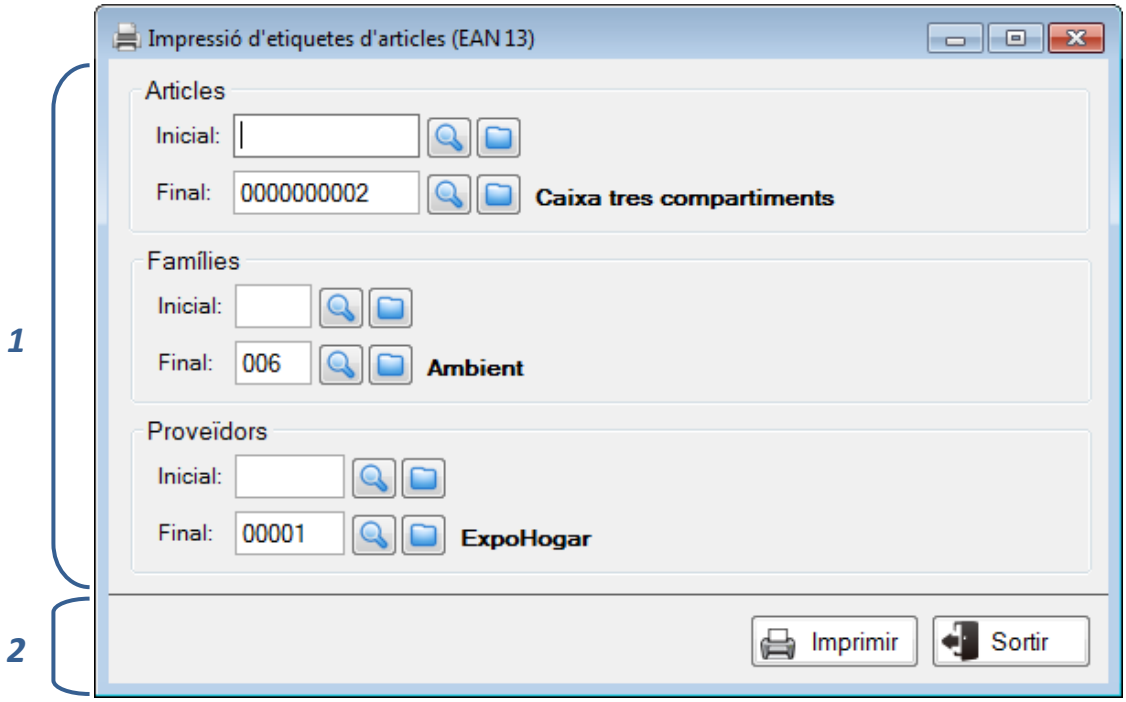
   Tenir en compte que fent clic a la capçalera de cada columna, ordenarem els registres per aquella columna i a més establirem la columna de cerca en aquesta.

3. **Peu:** hi tindrem els botons de "Seleccionar" i "Cancel·lar". Al pitjar el botó "Seleccionar", estarem indicant que volem capturar el registre que estigui actiu en aquell moment. També ho podrem fer pitjant directament la tecla "Enter".

9.3.6 – FORMULARI DE LLISTATS

Aquest tipus de formulari ens servirà per poder realitzar un filtre/selecció de les dades que volem enviar per la impressora. Cada filtre individual del llistat, es correspon a un camp contingut en les dades que ens disposem a llistar, i que podrem acotar-lo des d'un valor inicial a un de final per afinar el conjunt de dades a obtenir.

En la nostra aplicació únicament tenim dos formularis d'aquest tipus, el llistat de vendes i el d'impressió d'etiquetes d'articles que mostrem a continuació.



Il·lustració 18. Formulari de llistats

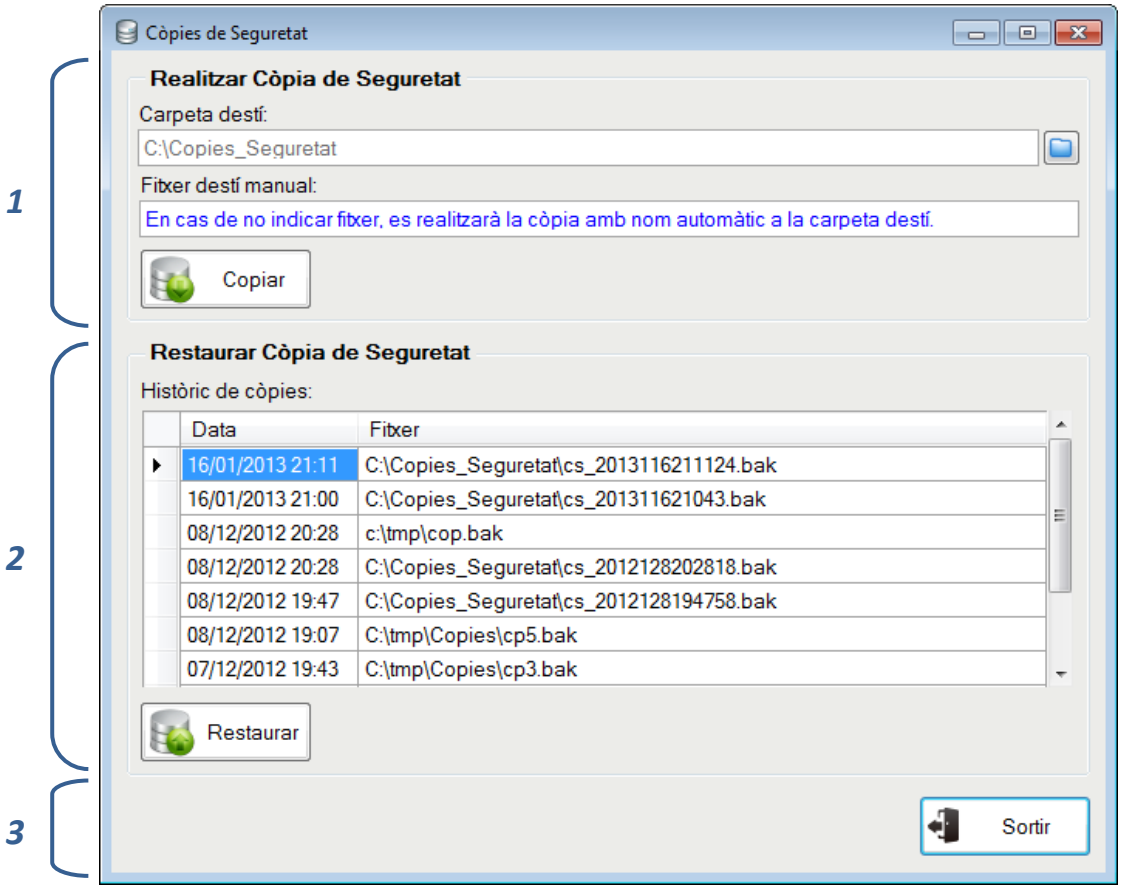
Estructura dels formularis de llistats:

1. En aquest apartat hi disposarem de tots els possibles filtres a indicar per l'obtenció de dades.

2. Conté els botons "Imprimir" i "Sortir". Al fer clic en el botó "Imprimir", es consultaran la informació a la base de dades tot aplicant els filtres escollits.

9.3.7 – FORMULARI DE GESTIÓ DE CÒPIES DE SEGURETAT

Utilitzarem el formulari de gestió de còpies de seguretat tant per realitzar còpies de seguretat com per restaurar les còpies de seguretat emmagatzemades.



Il·lustració 19. Formulari de gestió de còpies de seugretat

Tal com podem observar en la il·lustració 19, el formulari per la gestió de còpies de seguretat està dividit en 3 parts:

1. **Realitzar còpia de seguretat:** en aquesta part hi trobarem les diferents opcions per la realització de còpies de seguretat.
    a. Carpeta destí: caixa de text informativa que ens indica la carpeta per defecte on es realitzaran les còpies de seguretat, amb un botó per navegar-hi.
    b. Fitxer destí manual: caixa de text on hi podrem escriure la ruta i nom del fitxer on volem fer la còpia. En cas de no indicar-ne, es genera un nom automàtic.
    c. Botó "Copiar": botó per iniciar la còpia de seguretat.

2. **Restaurar còpia de seguretat:** conté una graella amb l'històric de totes les còpies de seguretat realitzades i el botó "Restaurar" que s'encarregarà de restaurar les dades de la còpia de seguretat escollida en la nostra aplicació.

3. **Botonera:** únicament hi ha el botó per sortir del formulari.

10 – IMPLEMENTACIÓ

La implementació d'una aplicació de software és una de les fases més rellevants, és on tot el treball previ de modelatge, d'anàlisi de requeriments, d'especificacions, d'estructuracions i de dissenys prenen fons i forma.

Per tal de dur a terme la fase d'implementació, haurem d'utilitzar eines de desenvolupament de software que incloguin algun llenguatge de programació. En el nostre cas, hem escollit l'eina Visual Studio 2010 de Microsoft, i dins d'aquest paquet el Visual C# com a llenguatge de programació. Podem veure en l'apartat 5.3.3 els motius pels quals hem pres aquesta decisió.

A continuació farem una petita descripció i repàs de les principals característiques de l'eina Visual Studio 2010 i del llenguatge de programació Visual C#.

10.1 – DESCRIPCIÓ I CARACTERTÍSTIQUES DE VISUAL STUDIO

Microsoft Visual Studio és un entorn de desenvolupament integrat per sistemes operatius Windows. Suporta varis llenguatges de programació com Visual C#, Visual C++, Visual J# i Visual Basic .NET, al igual que entorns de desenvolupament web com ASP .NET.

Visual Studio permet als desenvolupadors crear aplicacions d'escriptori, llocs i aplicacions web, així com serveis web en qualsevol entorn que suporti la plataforma .NET Framework[12].

Principals característiques de Visual Studio:

- Disposa d'un dels **millors entorns de desenvolupament integrats (IDE)**, que proporciona un conjunt d'eines destinades a ajudar a escriure i modificar el codi per als programes, així com detectar i corregir errors en els programes.

- **Intellisense** és una característica molt eficaç que pot augmentar de manera significativa la productivitat, ja que proporciona elements de codi lògics que es poden seleccionar en un menú desplegable quan s'escriu codi.

- Una altra característica interessant és la capacitat d'**especificar el Framework** sobre el que volem compilar la nostra aplicació. D'aquesta manera amb un simple canvi ja disposarem de la nostra aplicació preparada per diferents plataformes.

- Mitjançant **Visual Studio Tools per Office** podrem crear solucions en Visual Basic i Visual C# que ens permetran crear codi en documents Word i fulls d'Excel.

[12] És un component de software que pot ser o està inclòs en els sistemes operatius Microsoft Windows. Proveeix un extens conjunt de solucions predefinides per necessitats generals de la programació d'aplicacions, i gestiona l'execució dels programes escrits específicament per aquesta plataforma. Definició extreta de http://www.wikipedia.org.

- Inclou eines per al desenvolupament d'aplicacions per **dispositius intel·ligents** com els PDA i Smartphone.

- Totalment **compatible amb codi XML** que és una de les estructures optimitzades més utilitzades per la comunicació a través de Web. Inclou un dissenyador XML per facilitar la edició de XML i la creació d'esquemes XML.

- **Integració amb Visual Studio Team System** que és una plataforma d'eines de cicle de vida del desenvolupament de software extensible, integrat i productiu que ajuda als equips de desenvolupament de software mitjançant la millora de les comunicacions i la col·laboració durant tot el procés de desenvolupament. Inclou eines per al control de codi font, pel disseny d'aplicacions, pel control de qualitat, rendiment i testeig.

10.2 – DESCRIPCIÓ I CARACTERÍSTIQUES DE VISUAL C#

C# és un llenguatge de programació que ha estat dissenyat específicament per compilar aplicacions que s'executin sobre la plataforma .NET Framework. És un llenguatge simple, eficaç, amb seguretat de tipus i orientat a objectes. Les nombroses innovacions de C# permeten desenvolupar aplicacions ràpidament i mantenir la expressivitat i la elegància dels llenguatges de l'estil de C.

Visual C# és una implementació del llenguatge C# de Microsoft. Visual Studio ofereix compatibilitat amb Visual C# amb un complet editor de codi, un compilador, plantilles de projecte, dissenyadors, assistents de codi, un depurador eficaç i altres eines. La biblioteca de classes de .NET Framework ofereix accés a nombrosos serveis del sistema operatiu i altres classes útils que acceleraran el cicle de desenvolupament de manera significativa.

Principals característiques de Visual C#:

- És un llenguatge **orientat a objectes** pur, no admet ni funcions ni variables globals, tot el codi ha d'estar definit dins de classes amb el seu tipus de dades i accessibilitat. Això redueix al mínim els conflictes de noms i millora la llegibilitat del codi.

- Inclou la **seguretat de tipus de dades**, evitant errors de difícil detecció com podrien ser les conversions entre tipus de dades incompatibles, l'ús de variables sense inicialitzar, comprovació de paràmetres,...

- Utilitza un **sistema de tipus unificat**. En C# tots els tipus de dades són derivats de la classe base S*ystem.Object.* Això facilita la creació de col·leccions genèriques, ja que poden emmagatzemar objectes de qualsevol tipus.

- **Modernitat**. Inclou nous elements que al llarg del temps s'ha vist que són de gran utilitat i que en altres llenguatges s'han de simular. Un exemple podria ser la instrucció *foreach* que ens permet recórrer col·leccions d'objectes amb molta facilitat.

- La **orientació a components** possibilita la definició de propietats, "events" o atributs de forma senzilla.

- Incorpora un recol·lector de deixalles que ens proporciona una **gestió automàtica de la memòria**. Evita al desenvolupador d'haver de destruir els objectes utilitzats.

- Mitjançant la **extensibilitat d'operadors** es pot redefinir el significat per defecte de gran part dels operadors existents.

- L'**espai de noms** ajuda al desenvolupador a tenir ben organitzats els tipus de dades.

10.3 – ESTRUCTURA INTERNA DEL CODI

En la nostra aplicació hem aplicat una **arquitectura de tres capes** en la seva codificació, d'aquesta manera hem aconseguit separar les diferents lògiques presents. L'estructuració en capes ens pot aportar els següents beneficis:

- El desenvolupament es pot dur a terme en varis nivells.
- Desenvolupaments paral·lels en cada capa.
- Aplicacions més robustes degut a l'encapsulament.
- En cas que sorgeixi algun canvi, únicament caldrà atacar la capa afectada sense haver de revisar codis creuats.
- Manteniment i suport més senzill. És molt més senzill canviar un component que no pas modificar tota una aplicació d'una sola capa.
- Major flexibilitat. Ens permetrà afegir noves funcionalitats sense complicacions.

Tot seguit passem a descriure les tres capes implementades:

1. **Capa d'interfície**: aquesta capa és l'encarregada de presentar l'aplicació a l'usuari i interactuar amb ell. És interessant implementar aquesta capa el més entenedora i amigable possible, per tal que li sigui més fàcil a l'usuari d'entendre el funcionament i la lògica de l'aplicació. La capa d'interfície únicament es comunica amb la capa de negoci. En la nostra aplicació, hi tenim definits tots els formularis, controls visuals, reports d'impressió i imatges.

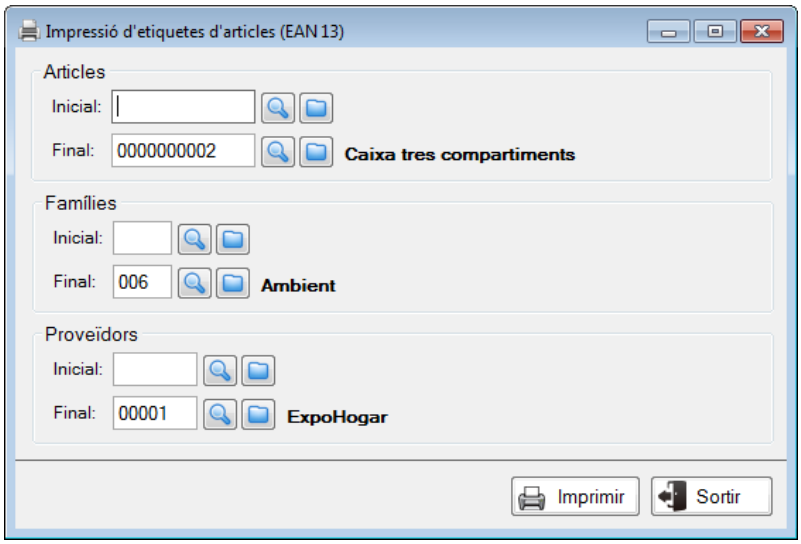
2. **Capa de negoci**: en la capa de negoci és on es reben les peticions de l'usuari i es retornen les respostes una vegada processades. Si estableixen les regles que s'han de complir. La capa de negoci es comunica amb la capa d'interfície per rebre les peticions dels usuaris i presentar els resultats, i amb la capa d'accés a dades per tal d'obtenir o emmagatzemar-hi les dades necessàries. En el nostre cas, hi hem implementat classes amb multitud d'operacions de preparació de consultes i actualitzacions, també amb operacions referents a les còpies de seguretat i als codis de barres EAN.

3. **Capa d'accés a dades**: és la capa responsable d'accedir a la base de dades de l'aplicació per tal de dur a terme les peticions de consulta o d'actualització d'informació. Únicament es comunica amb la capa de negoci. En la nostra implementació, hem definit una classe per accedir a la base de dades que ens permet realitzar les operacions habituals de consultes i actualitzacions de les dades. Cal anomenar també que s'ha realitzat de manera que ens **permeti accedir a bases de dades de diferents proveïdors** com SQL Server o MySql. La base de dades ha de ser igual en els diferents gestors de bases de dades on la vulguem definir.

A continuació i mitjançant un exemple real de la nostra aplicació, veurem la comunicació entre les diferents capes tot mostrant i explicant la seva codificació.

10.3.1 – CAPA D'INTERFÍCIE

En primer lloc i a través de l'opció adequada en el menú lateral de l'aplicació, l'usuari obrirà el formulari corresponent a la impressió d'etiquetes d'articles. En aquest punt apareix el formulari mostrat en la il·lustració número 20, el qual està inclòs en la capa d'interfície.



Il·lustració 20. Formulari d'impressió d'etiquetes (Capa d'interfície)

Una vegada elegits els filtres en el formulari d'impressió d'etiquetes, l'usuari pitja el botó "Imprimir" per tal d'enviar per impressora les etiquetes dels articles resultants dels filtres. En la il·lustració número 21 hi podem veure el codi que s'executarà en el moment de pitjar el botó "Imprimir", en concret es dispararà el mètode ***btImprimir_Click()*** corresponent a l'event *Click* del botó "Imprimir".

Explicació del mètode *btImprimir_Click()*:

Es tracta d'un mètode privat inclòs en el codi de la classe que implementa el formulari d'impressió d'etiquetes. Aquest mètode és de tipus **void,** això ens indica que realitzarà un seguit d'accions sense retornar cap valor.

Es crea i inicialitza les variables **lcFiltre** i **lcOrdre.** A la variable **lcFiltre** hi col·loquem els filtres obtinguts del formulari, i a la variable **lcOrdre** hi indiquem el nom del camp pel qual volem ordenar les dades a obtenir.

Per un altre costat creem l'objecte **objArt** mitjançant l'operador **new.** Aquest objecte representa una nova instància de la classe **article** que deriva de la classe base **taula** i es troba en la capa de negoci. La classe **article** representa un registre de la taula d'articles en la base de dades. A més a més ens proveeix de totes les operacions heretades de la classe **taula,** per realitzar diferents tipus de consultes i actualitzacions relacionades amb la taula d'articles de la base de dades.

Utilitzem el mètode **ObtenirDataTableSet()** a partir de l'objecte **objArt** per tal d'obtenir els articles corresponents als filtres indicats al formulari segons les variables **lcFiltre** i **lcOrdre**. Aquest mètode heretat de la classe **taula**, ens retorna a la variable privada de nivell de formulari **dtArticles** una estructura de dades de tipus **DataTable** amb tots els registres corresponents de la taula d'articles de la base de dades.

Acte seguit comprovem que hi hagi algun article segons els criteris de filtre escollits, en cas negatiu mostrem un missatge d'avís a l'usuari i retornem el control al formulari.

En aquest punt ja podem començar a preparar la impressió dels codis de barres EAN. Per tal propòsit crearem un objecte de tipus **PrintDocument,** que ens representa un document d'impressió. Anirem omplint aquest document amb els codis de barres generats a partir del mètode privat **Preparar_Eans()** del formulari d'impressió d'etiquetes.

```
private void btImprimir_Click(object sender, EventArgs e)
{
    article objArt = new article();
    string lcFiltre = "", lcOrdre = "";

    // Preparar el filtre segons les acotacions introduïdes per pantalla
    lcFiltre = "codi >= '" + txtArticleIni.txtBox.Text + "' AND codi <= '" + txtArticleFi.txtBox.Text + "' AND " +
               "familia >= '" + txtFamiliaIni.txtBox.Text + "' AND familia <= '" + txtFamiliaFi.txtBox.Text + "' AND " +
               "proveidor >= '" + txtProIni.txtBox.Text + "' AND proveidor <= '" + txtProFi.txtBox.Text + "' ";

    // Preparar ordre del llistat
    lcOrdre = "codi";

    // Obtenir els articles seleccionats
    dtArticles = (DataTable)objArt.ObtenirDataTableSet(false, lcFiltre, lcOrdre);

    // Comprobem que existeixi algun article segons els filtres
    if (dtArticles == null || dtArticles.Rows.Count <= 0)
    {
        MessageBox.Show("No existeix cap article segons els filtres escollits.", "Impressió d'etiquetes d'articles",
                        MessageBoxButtons.OK, MessageBoxIcon.Information);
        return;
    }

    try
    {
        // Preparar impressió etiquetes EAN 13
        PrintDocument pd = new PrintDocument();
        pd.PrintPage += new PrintPageEventHandler(this.Preparar_Eans);

        // Imprimir etiquetes
        pd.Print();
    }
    catch (Exception loEx)
    {
        MessageBox.Show("No s'ha pogut realitzar la impressió d'etiquetes degut al següent problema: "+Environment.NewLine +
                        loEx.Message, "Impressió d'etiquetes d'articles", MessageBoxButtons.OK, MessageBoxIcon.Error);
    }
}
```

Il·lustració 21. Codi botó "Imprimir" (Capa d'interfície)

Finalment i en cas que no hi hagi hagut cap incidència, enviem el document a la impressora predeterminada mitjançant la crida al mètode **Print()** de l'objecte **PrintDocument.** En cas de detectar alguna incidència mostrem un missatge d'avís a l'usuari

10.3.2 – CAPA DE NEGOCI

Seguint amb l'exemple anterior, hem vist com creàvem l'objecte **objArticle** corresponent a la classe **article** mitjançant la següent instrucció des de la capa d'interfície:

```
article objArt = new article();
```

En la il·lustració número 22 hi podem trobar la definició de la classe **article** que deriva de la classe base **taula,** i que es troben en la capa de negoci de la nostra aplicació. Com podem veure s'hi declaren tantes propietats públiques com camps té la taula article en la base de dades. Cal tenir en compte que en la classe base **taula,** s'hi declaren dues propietats públiques més anomenades **codi** i **nom** que heretarà la classe **article.**

També hi podem veure el constructor de classe, on es preparen les sentències per afegir, actualitzar o eliminar un article a la base de dades.

```
// ****************************************  INICI CLASSE DERIVADA ARTICLE  **************************************** //
public class article : taula
{
    // Definició de les propietat específiques de l'objecte article
    public string   familia     = string.Empty;
    public string   proveidor   = string.Empty;
    public decimal  tpcmarge    = decimal.Zero;
    public DateTime drevisio    = DateTime.Now;
    public decimal  cmcompra    = decimal.Zero;
    public string   foto        = string.Empty;
    public string   observa     = string.Empty;

    // Constructor de la classe "article" derivada de "taula"
    public article() : base()
    {
        // Estableixo el nom de la taula
        this.nomTaula = "article";

        // Defineixo les instruccions d'inserció, actualització i eliminació d'articles
        this.sqlInsert = "INSERT INTO article(codi, nom, familia, proveidor, tpcmarge, cmcompra, drevisio, foto, observa ) " +
                         "VALUES(@codi, @nom, @familia, @proveidor, @tpcmarge, @cmcompra, @drevisio, @foto, @observa) ";

        this.sqlUpdate = "UPDATE article " +
                         "SET nom=@nom, familia=@familia, proveidor=@proveidor, foto=@foto, observa=@observa, tpcmarge=@tpcmarge, " +
                             "cmcompra=@cmcompra, drevisio=@drevisio " +
                         "WHERE codi=@codi";

        this.sqlDelete = "DELETE FROM " + this.nomTaula + " WHERE codi=@codi";
    }

    // Redefinició del mètode AssignarParametres de la classe base
    public override void AssignarParametres(BaseDades db, bool blNomesClau)...

    // Redefinició del mètode Omplir de la classe base
    public override void Omplir(DataRow drValors)...

}
// ****************************************  FI CLASSE DERIVADA ARTICLE  **************************************** //
```

Il·lustració 22. Classe derivada "article" (Capa de negoci)

En la part final de la classe **article,** hi tenim un parell de redefinicions de mètodes de la classe base amb les peculiaritats de la classe **article.** La resta de mètodes encara que no estiguin definits aquí s'hereten de la classe base, com per exemple el mètode **ObtenirDataTableSet()** que utilitzem en la impressió d'etiquetes i que tot seguit descriurem.

En la il·lustració número 23 hi ha la codificació de les dos definicions que hem implementat del mètode **ObtenirDataTableSet()** en la classe base **taula.**

```
1) public object ObtenirDataTableSet(bool tlDataset, string tcFiltre, string tcOrdre)

2) public object ObtenirDataTableSet(bool tlDataset, string tcConsulta)
```

Les dos definicions es tracten de mètodes d'àmbit públic i que retornen un objecte de tipus **object.** Ho fem d'aquesta manera per poder retornar un objecte de tipus **DataTable** o **DataSet** segons el valor del paràmetre **tlDataset.**

En la primera definició, hi tenim dos paràmetres més (**tcFiltre** i **tcOrdre**) que ens serveixen per construir la instrucció SQL[13] per obtenir les dades desitjades de la base de dades. Passarem aquesta instrucció SQL per paràmetre a la segona definició del mètode **ObtenirDataTableSet().**

Amb aquestes dos definicions, possibilitem l'obtenció d'articles a partir d'uns filtres i un ordre o a partir d'una instrucció SQL complerta, tot unificant codi al màxim i podent retornar la informació en dos formats diferents.

```
// Ens crea un objecte DATATABLE o DATASET amb tots els registres de la taula segons el paràmetre tlDataset,
// a partir del filtre i ordre indicats als paràmetres tcFiltre i tcOrdre
public object ObtenirDataTableSet(bool tlDataset, string tcFiltre, string tcOrdre)
{
    string lcSql = "SELECT * FROM " + this.nomTaula;

    if (!string.IsNullOrEmpty(tcFiltre))
        lcSql = lcSql + " WHERE " + tcFiltre;

    if (!string.IsNullOrEmpty(tcOrdre))
        lcSql = lcSql + " ORDER BY " + tcOrdre;

    return ObtenirDataTableSet(tlDataset, lcSql);
}

// Ens crea un objecte DATATABLE o DATASET amb tots els registres de la taula segons el paràmetre tlDataset,
// a partir de la consulta sql indicada en el paràmetre tcConsulta
public object ObtenirDataTableSet(bool tlDataset, string tcConsulta)
{
    BaseDades db = new BaseDades();
    DbDataReader drTaula;
    DataSet dsTaula = new DataSet();
    DataTable dtTaula = new DataTable();

    // Conexió a la bd i execució de la consulta
    db.Conectar();
    db.CrearInstruccioSql(tcConsulta);
    drTaula = db.ExecutarConsultaSql();
    dtTaula = db.ConvertirDataReaderDataTable(drTaula, this.nomTaula);
    drTaula.Close();
    db.Desconectar();

    if (tlDataset)
    {
        // Si es demana un Dataset, afegeixo el datatable a un dataset
        dsTaula.Tables.Add(dtTaula);

        return dsTaula;
    }
    else
        return dtTaula;
}
```

Il·lustració 23. Classe base "taula" (Capa de negoci)

[13] El llenguatge de consulta estructurat o SQL (per les sigles en anglès *Structured Query Language*) és un llenguatge declaratiu d'accés a bases de dades relacionals que permet especificar varis tipus d'operacions. Definició extreta de http://www.wikipedia.org.

En la segona definició del mètode **ObtenirDataTableSet()** ens posem en contacte amb la capa d'accés a dades. En concret mitjançant la creació i utilització de l'objecte **db,** que contindrà una instància de la classe **BaseDades** definida en la capa d'accés a dades.

Des de la impressió d'etiquetes utilitzem la primera crida del mètode **ObtenirDataTableSet()** amb els paràmetres de filtres i ordre, que finalment acaba preparant una consulta SQL que es enviada a la capa d'accés a dades per obtenir els registres dels articles corresponents.

10.3.3 – CAPA D'ACCÉS A DADES

Continuant amb l'exemple de la impressió d'etiquetes, hem vist com des de la capa de negoci creàvem una instància de la classe **BaseDades** a través de la instrucció següent:

```
BaseDades db = new BaseDades();
```

En la il·lustració número 24 hi podem veure part de la definició de la classe **BaseDades** on hi definim varies propietats privades per al tractament de les connexions, transaccions, instruccions, proveïdor de dades, ...

```
// Classe que ens permetrà l'accés a una base de dades així com diferents operacions independentment del proveïdor de dades.
public class BaseDades
{

    #region "Propietats privades"

    private         DbConnection        objConexioBd       = null;
    private         DbCommand           objInstruccioSql   = null;
    private         DbTransaction       objTransaccioBd    = null;
    private static  DbProviderFactory   objProveidorBd     = null;
    private         string              strCadenaConexioBd = string.Empty;

    #endregion


    #region "Constructors"

    // Constructors per crear una instancia de la base de dades
    public BaseDades()
    {
        Configurar(false);
    }

    public BaseDades(bool tlMaster)
    {
        Configurar(tlMaster);
    }

    #endregion


    # region "Mètodes privats"

    // Mètode privat per configurar la base de dades
    private void Configurar(bool tlMaster=false)
    {
        try
        {
            string strProveidor = ConfigurationManager.AppSettings.Get("PROVEIDOR_DADES");
            objProveidorBd = DbProviderFactories.GetFactory(strProveidor);

            if (tlMaster)
                strCadenaConexioBd = ConfigurationManager.AppSettings.Get("CADENA_CONEXIO_MASTER");
            else
                strCadenaConexioBd = ConfigurationManager.AppSettings.Get("CADENA_CONEXIO");
        }
        catch (ConfigurationException objExcepcio)
        {
            throw new BaseDadesExcepcio("Error al obtenir la configuració del accés a dades.", objExcepcio);
        }
    }

    #endregion
```

Il·lustració 24. Definició classe "BaseDades" (Capa d'accés a dades)

Hi trobem la definició de dos constructors, un d'ells per al tractament normal de dades i l'altre per la gestió de les còpies de seguretat.

També hi ha el mètode privat **Configurar()** que ens ajuda a configurar la cadena de connexió a la base de dades segons el proveïdor de dades usat. També té en compte el paràmetre **tlMaster** per si ha de preparar la cadena de connexió en mode "còpies de seguretat".

En la il·lustració número 25 podeu veure alguns dels mètodes de la classe **BaseDades** utilitzats des de la capa de negoci.

```
// Mètode per conectar a la base de  dades
public void Conectar()
{
    if (this.objConexioBd != null && !this.objConexioBd.State.Equals(ConnectionState.Closed))
    {
        throw new BaseDadesExcepcio("La conexión ja està oberta.");
    }
    try
    {
        if (this.objConexioBd == null)
        {
            this.objConexioBd = objProveidorBd.CreateConnection();
            this.objConexioBd.ConnectionString = this.strCadenaConexioBd;
        }
        this.objConexioBd.Open();
    }
    catch (DataException excepcio)
    {
        throw new BaseDadesExcepcio("Error en la conexió a la base de dades.", excepcio);
    }
}


// Mètode per desconectar de la base de dades
public void Desconectar()
{
    if (this.objConexioBd.State.Equals(ConnectionState.Open))
    {
        this.objConexioBd.Close();
        this.objConexioBd.Dispose();
    }
}


// Mètode que ens crea i prepara una instrucció a la base de dades a partir d'una cadena SQL
// per la posterior execució
public void CrearInstruccioSql(string cadenaSQL)
{
    this.objInstruccioSql = objProveidorBd.CreateCommand();
    this.objInstruccioSql.Connection = this.objConexioBd;
    this.objInstruccioSql.CommandType = CommandType.Text;
    this.objInstruccioSql.CommandText = cadenaSQL;
    if (this.objTransaccioBd != null)
    {
        this.objInstruccioSql.Transaction = this.objTransaccioBd;
    }
}
```

Il·lustració 25. Mètodes classe "BaseDades" (Capa d'accés a dades)

11 – PROVES DEL SOFTWARE

La prova del software és un element d'un tema més ampli que moltes vegades es coneix com verificació i validació.

- La **verificació** es refereix al conjunt d'activitats que asseguren que el software implementa correctament una funció específica.

- La **validació** fa referència a un seguit d'accions que garanteixen que el software construït s'ajusta als requisits del client.

Les proves constitueixen l'últim escaló des del que es pot avaluar la qualitat del software i el descobriment d'errors, abans de la finalització del producte.

Per tal de garantir al màxim la qualitat del nostre software i reduir al mínim els errors, realitzarem les proves següents:

- Proves d'unitat
- Proves d'integració
- Proves de validació
- Proves de sistema

11.1 – PROVES D'UNITAT

Les proves d'unitat es centren en la verificació de la menor unitat del disseny del software: el mòdul. Un programa és un grup de mòduls que duen a terme una acció en concret.

Hem realitzat les proves d'unitat en la nostra aplicació tot verificant els següents punts:

- La **interfície** de cada mòdul, per assegurar que la informació flueix de forma adequada.

- Les **estructures de dades locals**, per garantir que les dades que es mantenen temporalment conserven la seva integritat durant tots els passos de l'execució.

- Les **condicions límit,** per assegurar que el mòdul funciona correctament en els límits establerts com restriccions de processament.

- Tots els **camins independents** de l'estructura de control, amb el fi de cobrir totes les sentències del mòdul. Que no quedi cap instrucció per executar.

- El **tractament d'errors**, per comprovar que efectivament els errors son interceptats.

11.2 – PROVES D'INTEGRACIÓ

De vegades ens podem trobar en que s'ha superat la fase de proves d'unitat satisfactòriament, i per tant, podem arribar a pensar que la nostra aplicació funciona correctament. Però de fet l'únic que podem garantir, és que tots els mòduls de la nostra aplicació funcionen correctament per separat. Això no vol dir que a mesura que anem combinant les funcionalitats de diferents mòduls puguin sorgir alguns inconvenients. Per cobrir aquesta necessitat, apareixen les proves d'integració.

S'ha comprovat que els enllaços entre els diferents mòduls del nostre software es satisfactori. A més a més també s'ha verificat que es mantenen les diferents funcionalitats dels mòduls després de ser enllaçats.

11.3 – PROVES DE VALIDACIÓ

Una vegada culminades les proves d'integració, el software està completament enllaçat com un sol paquet. S'han trobat i corregit tots els errors d'interfície, i ja es pot començar una sèrie de proves finals de validació.

Les proves de validació tenen per propòsit garantir que el software funciona d'acord a tots els requeriments del client. Aquests requeriments els hem especificat en l'apartat 5.1 de la fase d'anàlisi de requeriments. En el nostre cas hem validat els punts detallats a continuació.

- Les operacions d'alta, de baixa i de modificació en tots els formularis de manteniments.
- La gestió complerta de compres i vendes d'articles, realitzant documents nous, modificant documents existents i eliminant documents.
- La impressió d'etiquetes d'articles i del llistat de vendes utilitzant totes les opcions i filtres possibles.
- La realització i restauració de còpies de seguretat.

11.4 – PROVES DE SISTEMA

Les proves de sistema consisteixen en una sèrie de diferents proves on el propòsit primordial és exercitar profundament el sistema del ordinador. Han de verificar que s'han integrat adequadament tots els elements del sistema i que realitzen les funcions adequades.

A continuació passem a descriure les diferents proves de sistema que hem realitzat:

- **Prova de recuperació:** consisteix en forçar expressament la caiguda del sistema per observar si la recuperació de les dades es realitza de forma correcta. En el nostre cas,

vam desendollar l'ordinador de la corrent, i acte seguit vam engegar-lo i comprovar que les dades havien quedat intactes.

- **Prova de seguretat:** intenta verificar que els mecanismes de protecció incorporats en el sistema, el protegiran d'accessos indeguts. Hem comprovat en el nostre software, que no es possible accedir a l'aplicació obviant la pantalla d'autenticació, ni introduint un usuari i una clau incorrectes.

- **Prova de resistència:** es tracta d'executar l'aplicació en situacions extremes, on el sistema estigui consumint un nombre molt elevat de recursos. Es va executar la nostra aplicació satisfactòriament tenint obertes multitud d'aplicacions, que consumien una gran part dels recursos del sistema.

- **Prova de rendiment:** està dissenyada per provar el rendiment del software en temps d'execució dins del context d'un sistema integrat. És inacceptable el software que proporciona les funcions requerides però no s'ajusta als requisits de rendiment. Hem observat el rendiment del nostre software en diferents ordinadors que complien els requeriments mínims especificats en l'apartat 5.3.1. S'ha comprovat que el seu rendiment era òptim.

12 – CONSIDERACIONS, AMPLIACIONS I CONCLUSIONS

12.1 – CONSIDERACIONS

En aquest apartat comentarem alguns aspectes interessants de l'aplicació realitzada i que cal tenir en compte.

- Tot i que en l'apartat 5.3.3 expliquem l'elecció de Microsoft SQL Server per la gestió i emmagatzemament de la base de dades en el nostre software, **s'ha realitzat la capa d'accés a dades de tal manera que ens permeti utilitzar gran varietat de proveïdors de bases de dades**. Únicament modificant el fitxer de configuració xml de d'aplicació, on indicarem el proveïdor de dades a utilitzar i la cadena de connexió a la base de dades, ja podrem apuntar la nostra aplicació a una base de dades SQL Server, MySql, Oracle, Access,...

- Degut a la possibilitat de connectar la nostra aplicació amb diferents gestors de bases de dades, entenem que **el gestor de bases de dades no pot anar incorporat en la nostra aplicació** ni en el seu instal·lador. S'hauria d'estudiar el cas de cada client en concret per escollir el gestor de bases de dades que més s'adeqüi al seu negoci, i posteriorment en un procés independent de la nostra aplicació, instal·lar, configurar i adjuntar la base de dades per al nostre software.

- Mitjançant el mateix fitxer de configuració anomenat en el primer apartat, també podrem indicar quina és la caixa activa en cada terminal on hi hagi instal·lada l'aplicació. D'aquesta manera al entrar en el formulari del "**Document de venda**", ja ens **omplirà les dades de la capçalera automàticament** amb la caixa indicada al fitxer de configuració, i en conseqüència el venedor i client configurats al manteniment de la caixa.

- També cal destacar els **dos tipus d'usuaris** que hi pot haver en l'aplicació. Els usuaris de tipus **"Venedor"** que al entrar en l'aplicació accediran directament al formulari de vendes, i els usuaris de tipus **"Administrador"** que disposaran d'accés total a qualsevol part de l'aplicació.

12.2 – AMPLIACIONS

És molt interessant que una aplicació de gestió com la realitzada en aquest projecte, quedi oberta a possibles ampliacions i millores del producte. D'aquesta manera podrem garantir als clients una continuïtat del nostre software amb el seu creixement i expansió del negoci, podent satisfer noves necessitats i operatives.

A continuació descriurem algunes de les possibles ampliacions i millores que podríem realitzar en la nostra aplicació:

- Ampliar el document de venda amb el propòsit de poder realitzar **tiquets de venda** a més dels albarans de venda actuals, utilitzant numeracions independents per cada tipus de document.

- Relacionat amb l'ampliació del punt anterior, seria molt interessant afegir una nova opció per **l'arqueig de caixa**. Mitjançant aquesta nova opció, es podria comprovar tots els moviments de diners en cada caixa i verificar que els saldos quadrin al finalitzar la jornada laboral.

- Incorporar noves opcions per al **control d'estocs** a l'aplicació. A través d'aquestes opcions, l'usuari podria saber a cada moment de quantes unitats de cada article disposa, poden realitzar diversos llistats.

- En relació amb el punt anterior, també seria molt interessant agregar un **mòdul mòbil per la gestió d'inventaris**. Aquest mòdul o petita aplicació, es podria implementar per sistemes iOS o Android que ens permetria instal·lar-lo pràcticament en qualsevol dispositiu mòbil, ja sigui telèfon o tauleta. Per mitjà d'aquest mòdul l'usuari seria capaç de llegir codis de barres o codis BIDI[14], i directament amb el seu telèfon mòbil es desplaçaria pel magatzem o botiga i aniria llegint codis d'articles, per tal de generar un inventari de forma ràpida i amena. A més a més, també hi podríem afegir un apartat especial per al responsable de l'empresa, que li permetria controlar l'estoc dels articles des de casa seva i generar comandes per proveir articles sota mínims.

- Ampliar la manera actual de calcular el PVP dels articles, afegint **noves maneres de calcular el PVP**. Es podria obtenir a partir del preu mig de compra, del preu mig d'estoc, segons proveïdor,...

- Incloure un **mòdul de gestió i creació de llistats dinàmics**. Mitjançant aquest mòdul, l'usuari final disposaria de les eines adequades per tal de crear ell mateix tots els llistats que necessiti d'una manera intuïtiva i eficaç.

- Possibilitat d'afegir **nous documents de venda** com podrien ser les comandes o els pressuposts a clients.

- **Lligam amb comptabilitat**. Incorporar un mòdul de comptabilitat que ens permetria lligar les gestions portades en l'aplicació amb les conseqüents accions comptables.

- **Còpies de seguretat automàtiques**. Millorar la gestió de les còpies de seguretat permetent configurar-les i que es pugin realitzar automàticament cada cert lapse de temps.

[14] Es tracta de codis bidimensionals formats per quadrats blancs i negres que contenen informació codificada. La informació continguda pot ser un text, un enllaç a una pàgina web, etc. Definició extreta de http://www.gsmspain.com.

12.3 – CONCLUSIONS

Per un costat, podem afirmar que **els objectius establerts al inici del projecte s'han assolit amb solvència**. En l'apartat 2 d'aquesta documentació hi podem consultar els objectius a nivell general de la nostra aplicació, i en l'apartat 5.1 hi podem veure l'especificació d'aquests objectius convertits en requeriments funcionals.

Bé per ser estrictes, els dos últims punts de l'apartat 2 no s'han arribat a complir. Es tracta d'uns objectius que estan directament relacionats amb la presencia d'un client final. Aquest projecte no ha estat realitzat per cap client en concret, no s'ha arribat a instal·lar a cap client per la seva verificació, i per tant no els hem pogut complir.

Per altra banda, també puc confirmar que **he complert amb els objectius/motivacions personals** que m'havia fixat al començament del projecte. Un dels objectius personals més important per mi, era **l'aprenentatge d'un llenguatge de programació actual i totalment nou per a mi, el C# de la plataforma .NET de Microsoft**. Vaig creure importantíssim formar-me en aquest llenguatge de cara al meu futur en l'àmbit professional, ja que des que vaig sortir de la universitat i fins fa ben poc, havia estat absorbit pel llenguatge Visual FoxPro de Microsoft. Visual FoxPro és el llenguatge que utilitzem majoritàriament en l'empresa on treballo, un llenguatge en desús i ben aviat sense suport per part del fabricant. He de dir que abans de la finalització d'aquest treball final de carrera, en l'empresa on treballo em va sorgir la oportunitat de realitzar nous projectes precisament en aquest nou llenguatge, el Visual C#, per tant aquest objectiu personal va ser tot un èxit per mi abans d'hora i va ajudar a obrir-me portes i progressar professionalment.

Un altre objectiu personal, era el **realitzar una aplicació usant una arquitectura de tres capes**, separant les lògiques d'accés a dades, d'interfície i de negoci. També considerava bastant important dominar aquest estil de desenvolupament a dia d'avui, i que mitjançant l'elaboració d'aquest projecte he assolit.

La **utilització de l'eina Crystal Reports** en el disseny i generació dels informes en l'aplicació era un altre dels objectius personals que ha quedat superat.

Finalment també m'agradaria comentar que la realització d'aquest projecte ha estat un difícil i llarg camí per mi, degut a que vaig finalitzar la carrera fa nou anys (menys el TFC clar) i vaig començar a treballar al mateix temps. Ha estat molt gratificant veure com he anat superant cada obstacle i millorant poc a poc fins a la consecució d'aquest projecte.

13 – BIBLIOGRAFIA / WEBGRAFIA

13.1 – REFERÈNCIES BIBLIOGRÀFIQUES

[PRE02] Pressman, Roger. *Ingeniería del software – Un enfoque práctico.* 5ª Edició. McGraw-Hill. 2002.

[PRE92] Pressman, Roger. *Software Engineering, a practitioner's approach.* McGraw-Hill. 1992.

[BOE88] Boehm, Barry. *A Spiral Model for Software Development and Enhancement*. ACM SIGSOFT Software Engineering Notes. 1988.

[OUL99] Ould, Martin. *Managing Software Quality and Business Risk.* John Wiley & Sons. 1999.

[SHE89] Shepherd, A. *Analysis and training in information tasks.* Task Analysis for Human Computer Interaction. Chichester Ellis Horwood. 1989.

13.2 – LLIBRES I DOCUMENTACIONS CONSULTADES

Pressman, Roger. *Ingeniería del software – Un enfoque práctico.* 5ª Edició. McGraw-Hill. 2002.

Sommerville, Ian. *Ingeniería del software.* 7ª Edició. Addison Wesley. 2005.

Millera Codó, David. Memòria TFC "*Disseny d'una aplicació per a la gestió de la producció i consum en l'engreix de bestiar pel sistema integrat*". 2006.

Grau Vilas, Sergio. Memòria TFC "*Diseño e implementación de una aplicación de gestión para una empresa editorial*". 2007.

Roman Sabaté, Daniel. Memòria TFC "*Aplicació per a la gestió d'entrades i sortides de productes dins d'una empresa*". 2007.

Vidal Falcó, Yolanda. Memòria TFC "*Implantació d'un sistema ERP per a gestionar una empresa dedicada a la producció i distribució de material odontològic*". 2008.

Pérez Ortiz, Ivan. Memòria TFC "*Implementació d'una aplicació per a la gestió de pressupostos d'una botiga de mobles*". 2010.

Argilés Pons, Francesc. Memòria TFM "*Xarxes socials. El canal per al mainstream de les estrelles de la música*". 2012.

13.3 – WEBGRAFIA

Terminal punto de venta. *Wikipedia*. Definició general del concepte *Terminal Punt de Venda.* Darrera actualització el 15 d'abril de 2013.
http://es.wikipedia.org/wiki/Terminal_punto_de_venta

European Article Number (EAN). *Wikipedia*. Definició del famós sistema de codis de barres *EAN.* Darrera actualització el 7 de març de 2013.
http://es.wikipedia.org/wiki/European_Article_Number

Brainstorming. *Wikipedia*. Definició del mot Brainstorming. Darrera actualització el 21 d'abril de 2013.
http://es.wikipedia.org/wiki/Brainstorming

DIA. Pàgina web en anglès del software DIA que hem utilitzat per dissenyar el model entitat-relació. Darrera actualització el 23 de desembre de 2012.
https://live.gnome.org/Dia

Microsoft .NET Framework. *Wikipedia*. Descripció de la popular plataforma de Microsoft. Darrera actualització el 26 d'abril de 2013.
http://es.wikipedia.org/wiki/.NET_framework

SQL. *Wikipedia*. Definició del concepte SQL segons els usuaris. Darrera actualització el 26 d'abril de 2013.
http://es.wikipedia.org/wiki/SQL

BIDI. *Gsmspain*. Definició del sistema de codis BIDI. Darrera actualització el 22 de maig de 2013.
http://www.gsmspain.com/glosario/?palabra=BIDI

ANNEXOS

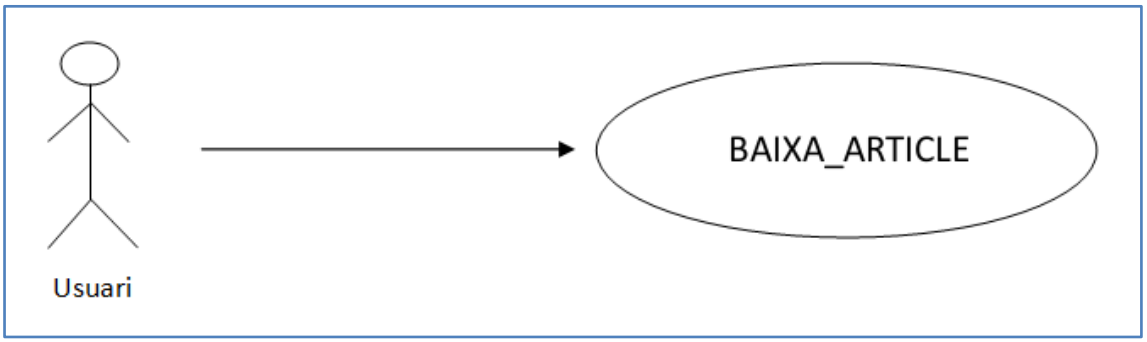
ANNEX 1 - CASOS D'ÚS

Cas d'ús 4. Baixa d'article / família / client / proveïdor / venedor / caixa TPV / usuari

Represento tots els processos de baixa en un sol cas d'ús ja que la operativa a realitzar és la mateixa.

En aquest cas d'ús es dóna de baixa un article / família / client / proveïdor / venedor / caixa TPV / usuari

Cas d'ús "Baixa_article"



Il·lustració 25. Cas d'ús "Baixa_article"

Flux d'esdeveniments "Baixa_article"

Usuari	Sistema
1.L'usuari vol donar de baixa un article, pitja el botó "Articles".	
	2.El sistema mostra el formulari per al manteniment d'articles inhabilitat amb diferents botons per les accions a realitzar.
3.L'usuari pitja el botó amb icona en forma de lupa o la tecla especial F3 per localitzar l'article que vol donar de baixa.	
	4.El sistema mostra una llista amb tots els articles existents amb diferents filtres per localitzar l'article ràpidament.
5.L'usuari selecciona l'article desitjat i pitja el botó "Seleccionar".	
	6.El sistema mostra al formulari d'articles les dades de l'article seleccionat.

7.L'usuari pitja el botó "Eliminar"	
	8.El sistema mostra un missatge de confirmació.
9.L'usuari confirma la eliminació de l'article.	
	10.El sistema elimina l'article de la base de dades.

Taula 24. Flux d'esdeveniments "Baixa_article"

PRE: {L'usuari vol eliminar un article ja existent al sistema.}

POST: {S'ha eliminat l'article.}

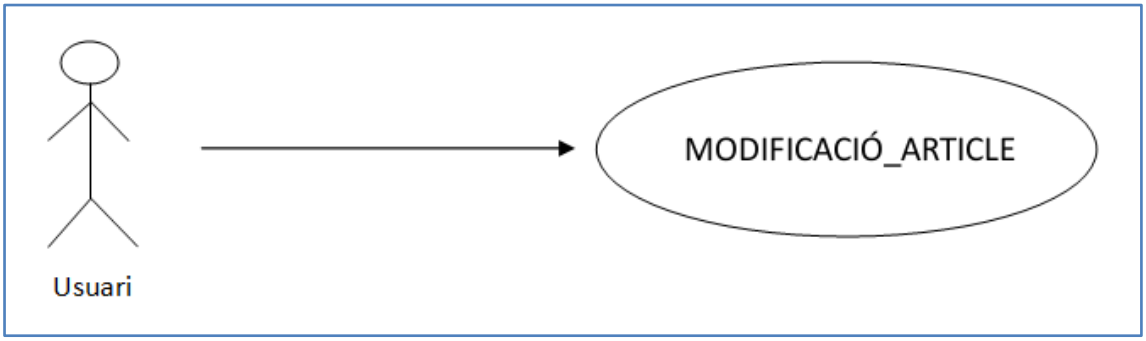
EXCEPCIÓ: {L'article està sent utilitzat pel sistema i no es pot eliminar. Es mostrarà missatge.}

Cas d'ús 5. Modificació d'article / família / client / proveïdor / venedor / caixa TPV / usuari

Per tal de mostrar tots els processos de modificació, els he englobat en un sol cas d'ús, ja que utilitzen la mateixa dinàmica de treball.

En aquest cas d'ús s'efectua la modificació d'un article / família / client / proveïdor / venedor / caixa TPV / usuari

Cas d'ús "Modificació_article"



Il·lustració 26. Cas d'ús "Modificació_article"

Flux d'esdeveniments "Modificació_article"

Usuari	Sistema
1.L'usuari vol modificar algun atribut de l'article, pitja el botó "Articles".	
	2.El sistema mostra el formulari per al manteniment d'articles inhabilitat amb diferents botons per les accions a realitzar.

3.L'usuari pitja el botó amb la icona en forma de lupa o la tecla especial F3 per localitzar l'article que vol modificar.	
	4.El sistema mostra una llista amb tots els articles existents amb diferents filtres per localitzar l'article ràpidament.
5.L'usuari selecciona l'article desitjat i pitja el botó "Seleccionar".	
	6.El sistema mostra al formulari els atributs de l'article seleccionat.
7.L'usuari pitja el botó "Editar"	
	8.El sistema habilita totes les caselles del formulari excepte el codi per a que l'usuari pugui modificar els atributs que desitgi.
9.L'usuari modifica els atributs desitjats i pitja el botó "Acceptar"	
	10.El sistema guarda els canvis realitzats en l'article a la base de dades.

Taula 25. Flux d'esdeveniments "Modificació_article"

PRE: {L'usuari vol modificar algun atribut/s d'un article ja existent al sistema.}

POST: {S'han modificat els atributs de l'article.}

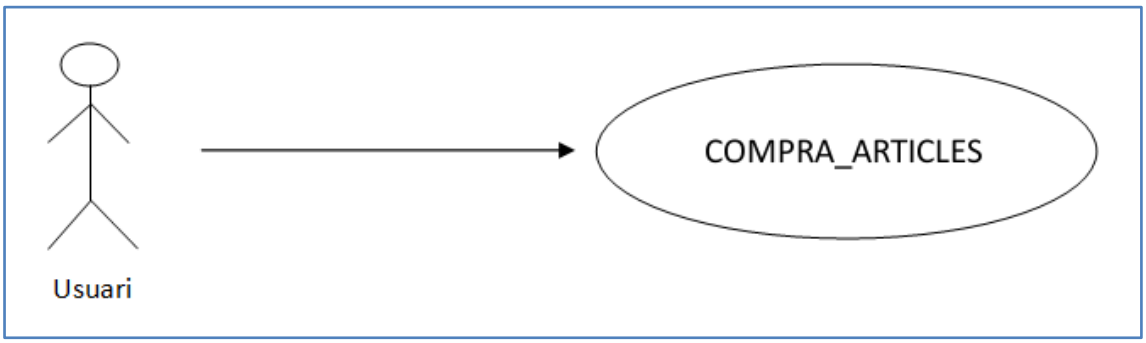
EXCEPCIÓ: {}

Cas d'ús 6. Compra d'articles

En aquest cas d'ús es realitza la compra d'articles que ens servirà per proveir el nostre magatzem i saber quan, a qui i a quin preu hem comprat els articles de que disposem.

Els atributs d'una compra són el codi de proveïdor, el número de document, la data i els diferents codis d'articles amb les seves unitats, preus i descomptes respectius.

Cas d'ús "Compra_articles"



Il·lustració 27. Cas d'ús "Compra_articles"

Flux d'esdeveniments "Compra_articles"

Usuari	Sistema
1.L'usuari vol realitzar la compra d'articles per omplir el magatzem, pitja el botó "Compres"	
	2.El sistema mostra el formulari per a la compra d'articles amb un nou document de compra apunt per la introducció de dades. El número de document i la data s'han omplert automàticament.
3.L'usuari introdueix el proveïdor a qui estem comprant, i modifica la data en la capçalera del document si ho desitja. Una vegada omplerta la capçalera, l'usuari va introduint les diferents línies del detall del document amb el codi d'article, les unitats, el preu de compra i els descomptes corresponents.	
	4.El sistema calcularà l'import a cada línia i el total del document.
5.En el moment que l'usuari hagi introduït totes les línies del document i cregui que el document ja està complert, pitjarà el botó "Acceptar". En cas que volgués desfer tot el document pitjaria el botó "Cancel·lar".	
	6.El sistema fa les comprovacions pertinents i en cas d'haver-hi alguna dada incorrecta ens mostrarà missatge d'avís.
	7.En cas que les comprovacions siguin correctes, el sistema emmagatzemarà el document introduït a la base de dades.

Taula 26. Flux d'esdeveniments "Compra_articles"

PRE: {L'usuari vol comprar articles a un proveïdor. Els articles i el proveïdor han d'existir a la base de dades}

POST: {S'ha generat un nou document de compra}

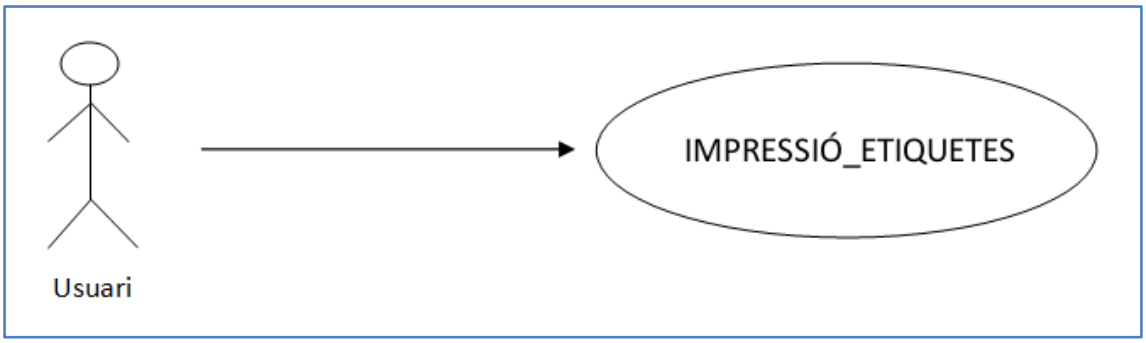
EXCEPCIÓ: { }

Cas d'ús 7. <u>Impressió d'etiquetes d'articles</u>

En aquest cas d'ús mostrem la impressió de les etiquetes dels articles amb els seus codis de barres.

Serà de molta utilitat per l'usuari poder etiquetar els articles amb codis de barres i així al vendre'ls només llegint el codi de barres amb el lector ja ens afegirà la venda automàticament.

Cas d'ús "Impressió_etiquetes"



Il·lustració 28. Cas d'ús "Impressió_etiquetes"

Flux d'esdeveniments "Impressió_etiquetes"

Usuari	Sistema
1.L'usuari vol realitzar una impressió d'etiquetes d'articles, pitja el botó "Etiquetes d'articles"	
	2.El sistema mostra el formulari per la impressió d'etiquetes amb la possibilitat de filtrar els articles per codis d'articles, famílies o proveïdors.
3.L'usuari introdueix els filtres desitjats i pitja el botó "Imprimir".	
	4.El sistema consulta la informació a la base de dades i envia les etiquetes dels articles resultants per impressora.

Taula 27. Flux d'esdeveniments "Impressió_etiquetes"

PRE: {L'usuari vol realitzar una impressió d'etiquetes d'articles.}

POST: {S'ha realitzat una impressió d'etiquetes d'articles.}

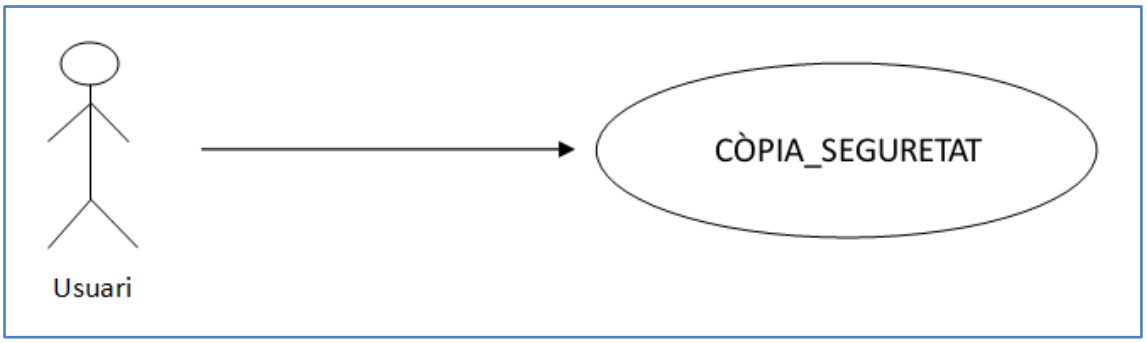
EXCEPCIÓ: {No hi ha articles a mostrar segons els filtres escollits per l'usuari. Errors relacionats amb la impressora.}

Cas d'ús 8. Còpia de seguretat

En aquest cas d'ús és realitza la còpia de seguretat de totes les dades de la base de dades.

Serà imprescindible la realització de còpies de seguretat per garantir les dades a l'usuari prevenint la pèrdua de dades per varietat de circumstàncies.

Cas d'ús "Còpia_seguretat"



Il·lustració 29. Cas d'ús "Còpia_seguretat"

Flux d'esdeveniments "Còpia_seguretat"

Usuari	Sistema
1.L'usuari vol realitzar una còpia de seguretat de totes les seves dades, pitja el botó "Còpies de seguretat"	
	2.El sistema mostra el formulari per la gestió de les còpies de seguretat amb la possibilitat de seleccionar la ruta i el nom del fitxer que contindrà la còpia.
3.L'usuari introdueix la ruta i el nom del fitxer desitjat i pitja el botó "Copiar".	
4.L'usuari pitja directament el botó "Copiar". D'aquesta manera es realitzarà la còpia de seguretat a la carpeta per defecte.	
	5.El sistema mostra un missatge de confirmació.
6.L'usuari confirma la realització de la còpia de seguretat.	
	7.El sistema realitza la còpia de seguretat de tota la base de dades i ho guarda en la ruta i fitxer escollits. També enregistra aquest fet en l'històric de còpies a la base de dades.

Taula 28. Flux d'esdeveniments "Còpia_seguretat"

PRE: {L'usuari vol realitzar una còpia de seguretat.}

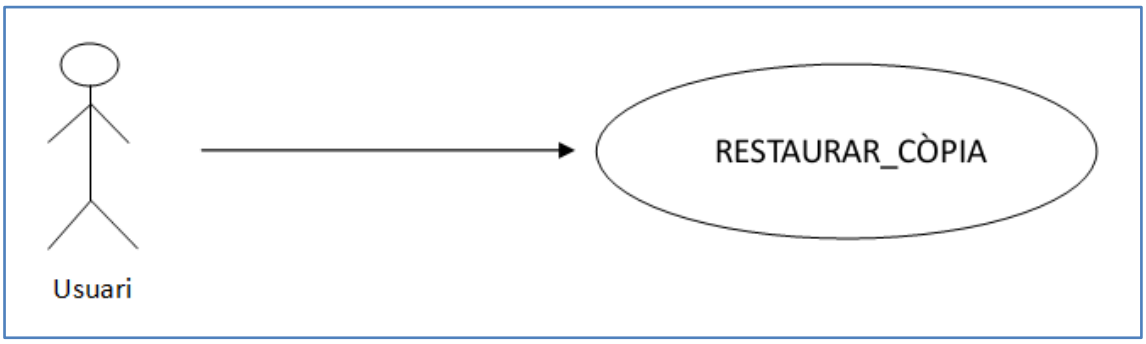
POST: {S'ha realitzat una còpia de seguretat.}

EXCEPCIÓ: {Diferents errors relacionats amb l'accés a disc i amb el servidor de la base de dades.}

Cas d'ús 9. Restaurar còpia de seguretat

En aquest cas d'ús és realitza la restauració d'una còpia de seguretat realitzada anteriorment pel l'usuari.

Cas d'ús "Restaurar_còpia"



Il·lustració 30. Cas d'ús "Restaurar_còpia"

Flux d'esdeveniments "Restaurar_còpia"

Usuari	Sistema
1.L'usuari vol restaurar una còpia de seguretat de totes les seves dades ja que per motius externs ha perdut dades, pitja el botó "Còpies de seguretat"	
	2.El sistema mostra el formulari per la gestió de les còpies de seguretat, on podem veure una llista amb l'històric de les còpies de seguretat realitzades. La llista apareix amb ordre descendent per la data de la còpia.
3.L'usuari selecciona la còpia de seguretat que vol restaurar i pitja el botó "Restaurar".	
	4.El sistema mostra un missatge de confirmació avisant del perill que suposa restaurar una còpia de seguretat.
5.L'usuari confirma que vol restaurar la còpia.	
	6.El sistema recupera tota la informació de la còpia de seguretat escollida i la col·loca a la base de dades.

Taula 29. Flux d'esdeveniments "Restaurar_còpia"

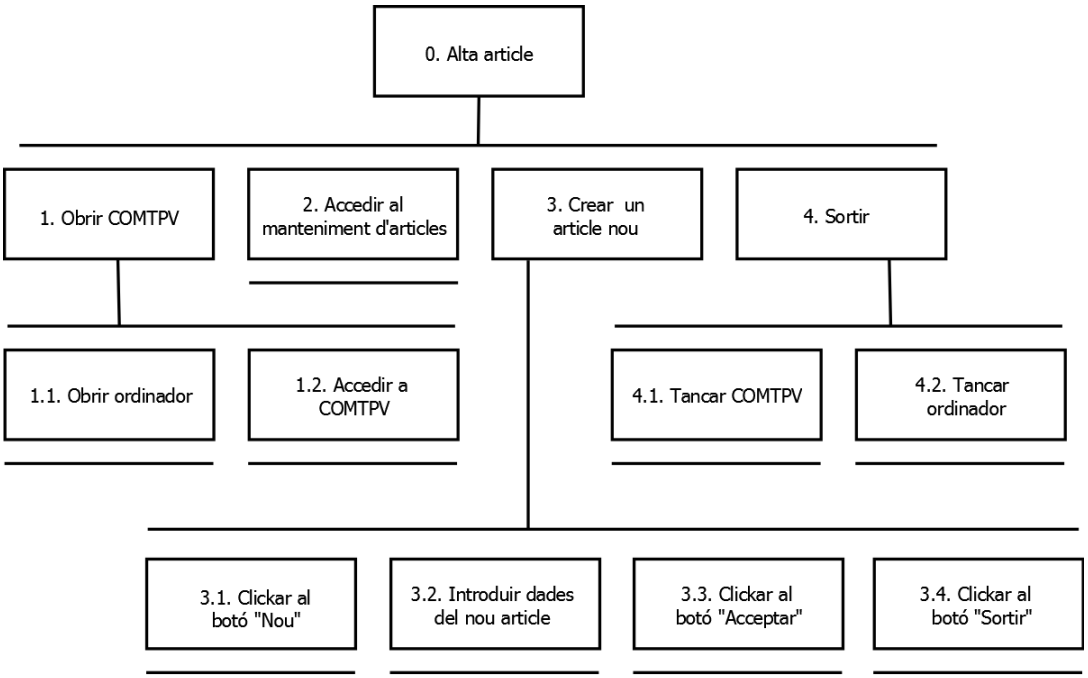
PRE: {L'usuari vol restaurar una còpia de seguretat.}

POST: {S'ha restaurat una còpia de seguretat.}

EXCEPCIÓ: {El fitxer de còpia de seguretat a restaurat no existeix. Diversos errors relacionats amb el servidor de base de dades.}

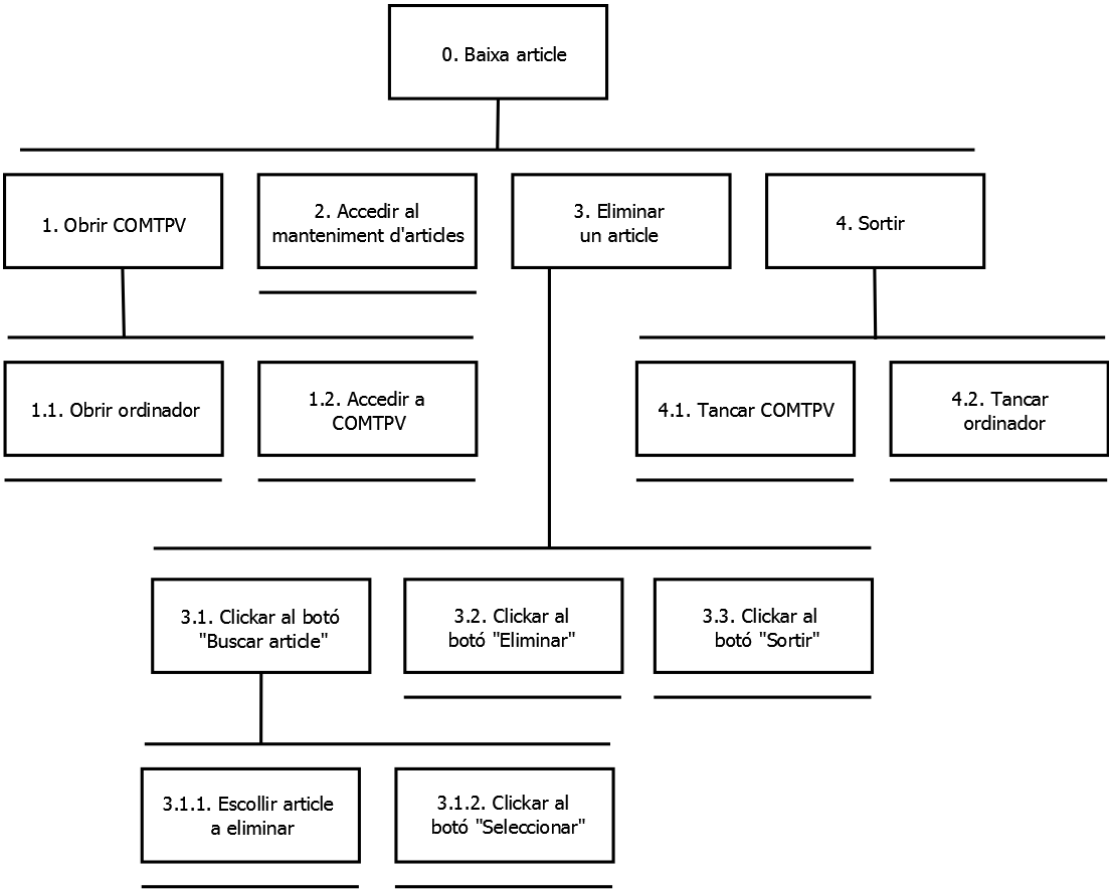
ANNEX 2 – ANÀLISI JERÀRQUIC DE TASQUES

Tasca 4. Alta d'article / família / client / proveïdor / venedor / caixa TPV / usuari



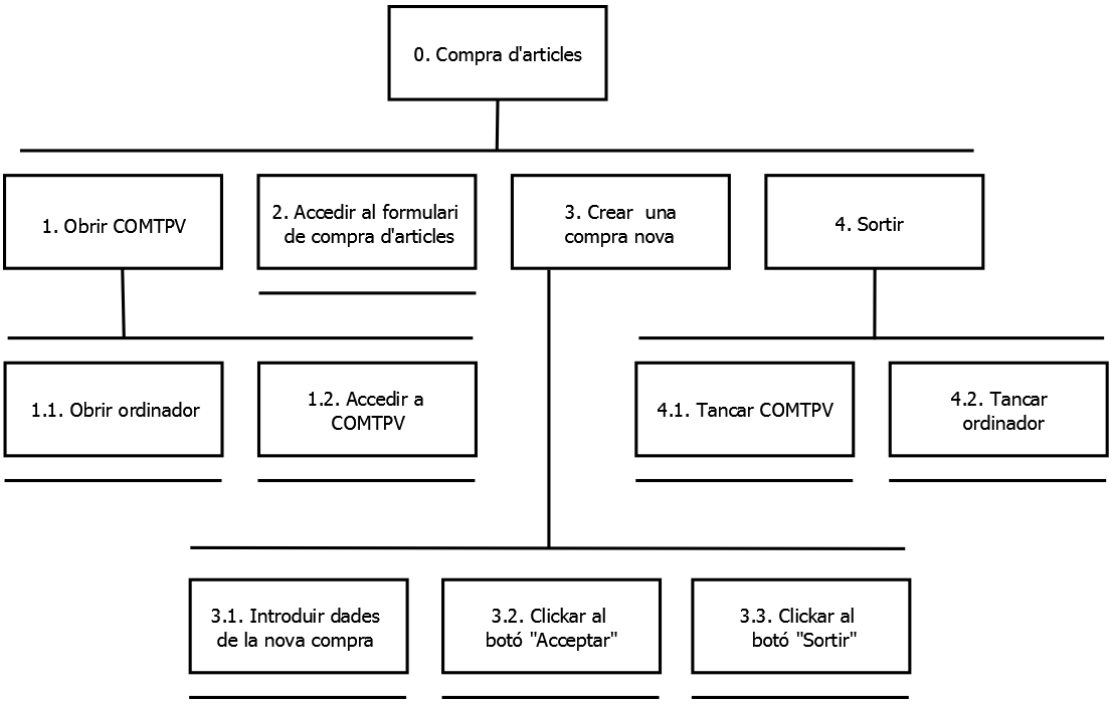
Il·lustració 31. Tasca "Alta article"

Tasca 5. Baixa d'article / família / client / proveïdor / venedor / caixa TPV / usuari



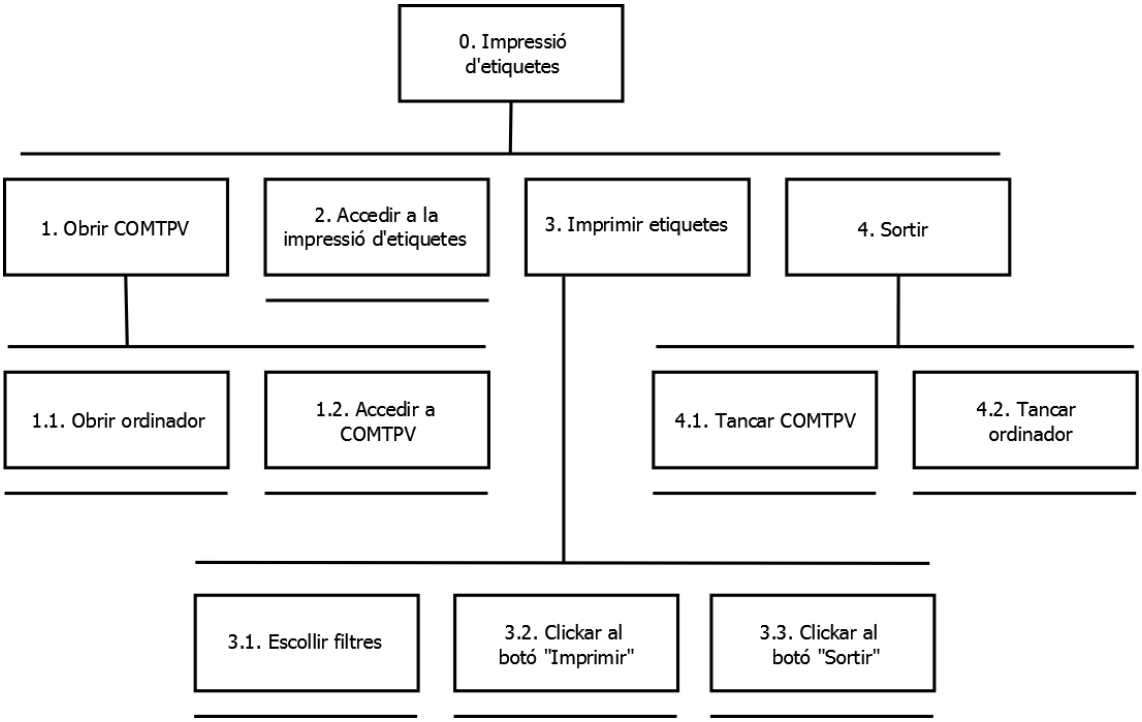
Il·lustració 32. Tasca "Baixa article"

Tasca 6. Compra d'articles



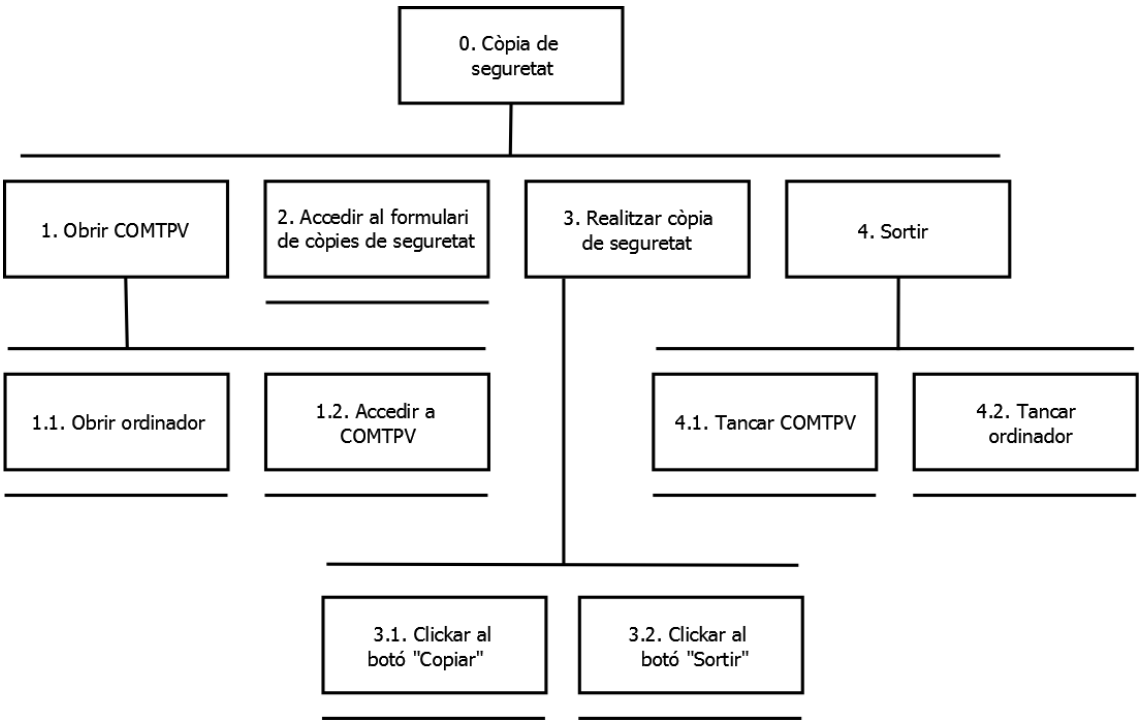
Il·lustració 33. Tasca "Compra d'articles"

Tasca 7. Impressió d'etiquetes d'articles



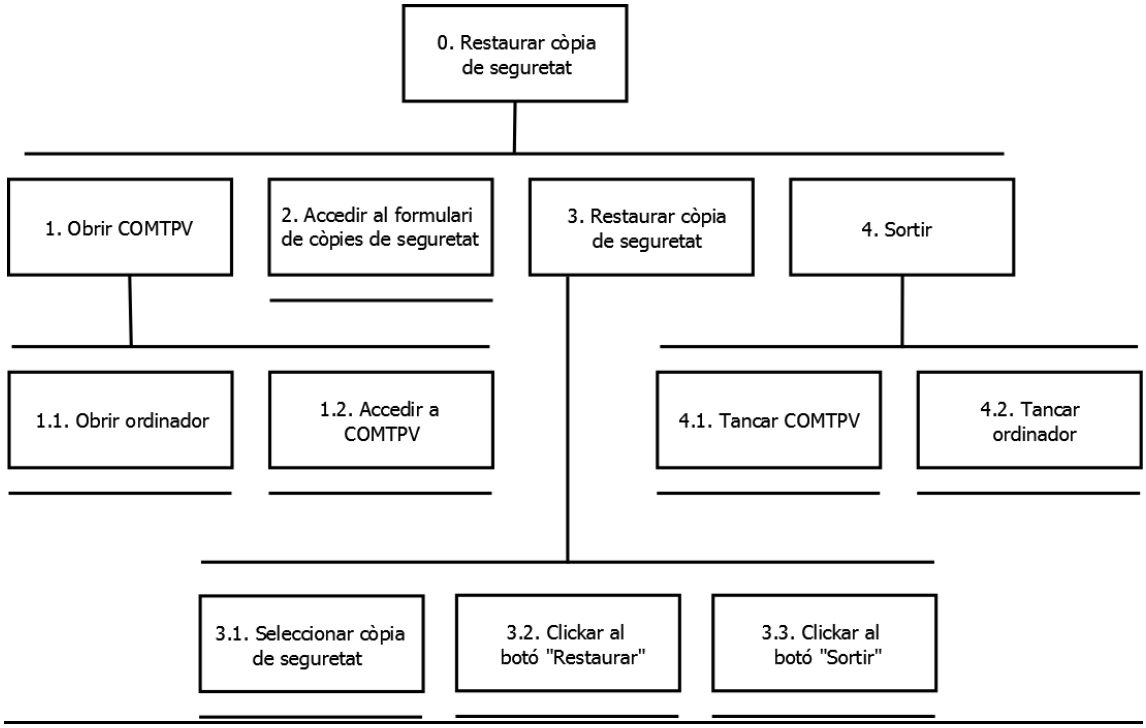
Il·lustració 34. Tasca "Impressió d'etiquetes d'articles"

Tasca 8. Còpia de seguretat



Il·lustració 35. Tasca "Còpia de seguretat"

Tasca 9. Restaurar còpia de seguretat



Il·lustració 36. Tasca "Restaurar còpia de seguretat"